AISTATS 2005

Proceedings of the
Tenth International Workshop on Artificial Intelligence and Statistics

Edited by

Robert Cowell
City University, London

Zoubin Ghahramani
University College London

January 6-8, 2005
The Savannah Hotel, Barbados

Published by *The Society for Artificial Intelligence and Statistics*

The web site for the AISTATS 2005 workshop may be found at

http://www.gatsby.ucl.ac.uk/aistats/

from which individual papers in these proceedings may be downloaded, together with a bibtex file with details all of the workshop papers. Citations of articles that appear in the proceedings should adhere to the format of the following example:

References

[1] Shivani Agarwal, Sariel Har-Peled, and Dan Roth. A uniform convergence bound for the area under the ROC curve. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, Jan 6-8, 2005, Savannah Hotel, Barbados*, pages 1–8. Society for Artificial Intelligence and Statistics, 2005. (Available electronically at http://www.gatsby.ucl.ac.uk/aistats/)

ISBN 0-9727358-1-X

Contents

Preface

The Society for Artificial Intelligence and Statistics (SAIAS) is dedicated to facilitating interactions between researchers in AI and Statistics. The primary responsibility of the society is to organize the biennial International Workshops on Artificial Intelligence and Statistics, as well as maintain the AI-Stats home page and mailing list on the Internet. The tenth such meeting took place in January 2005 in Barbados, a new venue for this conference and the first time it had been held outside of the United States of America. Details about the conference may be found at http://www.gatsby.ucl.ac.uk/aistats/.

Papers from a large number of different areas at the interface of statistics and AI were presented at the workshop. In addition to traditional areas of strength at AISTATS, such as probabilistic graphical models and approximate inference algorithms, the workshop also benefitted from high quality presentations in a broader set of new topics, such as semi-supervised learning, kernel methods, spectral learning, dimensionality reduction, and learning theory, to name a few. This diversity contributed to a strong and stimulating programme.

A novel feature of this workshop were prizes awarded to students for Best Student Papers. Three awards were made to Francis Bach, Philip J. Cowans and Kilian Weinberger. This was made possible by a donation from the NITCA at the Australian National University.

There were approximately 150 submissions. Almost every paper was assigned three reviewers, other than the program chairs. On the basis of these reviews, 21 papers were selected for presentation in the plenary session and 36 were selected for the poster sessions, based on their interest and relevance to the conference and on their originality and clarity of exposition. In deciding on which papers to accept we drew heavily on the reviews of the program committee. The standard was rigorous and impartial, and we note in passing that several members of the program committee had papers rejected.

The United States was the country with the most submissions (57) with Canada (17) and the United Kingdom (14) being the next largest contributing countries. Most of the remaining submissions came from Europe, with submissions from Belgium, Denmark, Finland, France, Germany, Ireland, Italy, Slovakia, Slovenia, Spain, Switzerland and The Netherlands. Papers submitted from outside of Europe and North America originated from Algeria, Australia, Brazil, Chile, Hong Kong, Iran, Israel, Japan, Malaysia and Russia. This range of countries emphasizes the truly international character of the conference. It is worth noting that several papers had their co-authors based in different countries.

Equal acceptance criteria were used for all submissions, and our decision of how each paper was presented was aimed at creating a varied programme rather than drawing a distinction between poster papers and plenary papers. Accordingly, in these proceedings we have ordered all of the papers alphabetically by the lead author's name. The conference web page may be consulted to see how individual papers were presented.

Robert Cowell and Zoubin Ghahramani, Program Chairs

Acknowledgements

The volume would not exist without the help of many people: the authors, the reviewers, the participants, the sponsors, and the Society for Artificial Intelligence and Statistics.

We should like to thank our invited speakers: Craig Boutilier, of the Department of Computer Science University of Toronto, who talked about "Regret-based Methods for Decision Making and Preference Elicitation"; Nir Friedman, of the School of Computer Science and Engineering, Hebrew University, who described "Probabilistic Models for Identifying Regulation Networks: From Qualitative to Quantitative Models"; Tom Minka, of Microsoft Research (Cambridge, UK), who spoke about "Some Intuitions About Message Passing", and Steffen Lauritzen, of the Department of Statistics at the University of Oxford, who talked about "Identification and Separation of DNA Mixtures using Peak Area Information". Regrettably, our fifth invited speaker, Tommi Jaakkola of the MIT Computer Science and Artificial Intelligence Laboratory was unable to attend the meeting.

We are also very grateful to all authors: authors of plenary papers, poster papers and those authors whose papers, though submitted, were not included in the conference program, which resulted in work of such a high standard.

We would like to thank our sponsors, NITCA at the Australian National University for the prize money for the Best Student Paper awards, and also Microsoft Research and the European PASCAL initiative for donations towards the running costs of the conference.

We would also like to thank Mrs. Faye Wharton-Parris of Premier Events, who took care among other things of local arrangements such as collecting registration fees, organizing accommodation and transportation between hotels, multimedia hire and wireless networking within the conference room. Her company's professional services ensured that the day-to-day running of the conference went smoothly.

Special thanks also to Katherine Heller at the Gatsby Unit, who designed and maintained the website, and helped with numerous aspects of the conference organization; to Daryl Pregibon at Google who handled finances for the deposits; and to Chani Johnson and others of Microsoft Research, who maintained the Conference Management Toolkit with which the reviewing process was coordinated.

Finally, we would especially like to thank the members of the program committee and a few external reviewers for agreeing to give up their time to review papers. The thorough and conscientious reviews of their allocated papers, which they carried out in a short period of time, ensured a high standard of presentations for the conference. The members of the program committee are listed on the following page.

Program Committee

Yasemin Altun, Brown University
Hagai Attias, Golden Metallic, Inc.
Francis Bach, University of California, Berkeley
Matthew Beal, SUNY Buffalo
Yoshua Bengio, University of Montreal
Christopher Bishop, Microsoft Research
David Blei, University of California, Berkeley
Olivier Bousquet, Pertinence
Wray Buntine, HIIT
Olivier Chapelle, Max Planck Institute
Guido Consonni, University of Pavia
Greg Cooper, University of Pittsburgh
Adrian Corduneanu, MIT
Denver Dash, Intel Research
Phil Dawid, University College London
Nando de Freitas, University of British Columbia
Vanessa Didelez, University College London
Michael Duff, Trinity College
Nir Friedman, Hebrew University of Jerusalem
Dan Geiger, Technion - Israel Inst. of Technology
Lise Getoor, University of Maryland
Paolo Giudici, University of Pavia
Tom Griffiths, Stanford University
Peter Grunwald, CWI, Netherlands
Carlos Guestrin, Carnegie Mellon University
David Heckerman, Microsoft Research
Ralf Herbrich, Microsoft Research
Tom Heskes, University of Nijmegen
Thomas Hofmann, Brown University
Chris Holmes, University of Oxford
Tommi Jaakkola, MIT
Nebojsa Jojic, Microsoft Research
Anitha Kannan, University of Toronto
Uffe Kjærulff, Aalborg University
John Lafferty, Carnegie Mellon University
John Langford, TTI - Chicago
Neil Lawrence, University of Sheffield
Guy Lebanon, Carnegie Mellon University
Juan Lin, Rutgers University
David Madigan, Rutgers University
Chris Meek, Microsoft Research
Marina Meilă, University of Washington
Tom Minka, Microsoft Research
Quaid Morris, University of Toronto
Iain Murray, Gatsby Unit
Kevin Murphy, MIT
Petri Myllymäki, University of Helsinki
Nuria Oliver, Microsoft Research
Manfred Opper, University of Southampton
Fernando Pérez-Cruz, Gatsby Unit
John Platt, Microsoft Research
Thomas Richardson, University of Washington
Imre Risi Kondor, Columbia University
Michal Rosen-Zvi, University of California, Irvine
Sam Roweis, University of Toronto
Lawrence Saul, University of Pennsylvania
Dale Schuurmans, University of Alberta
Paola Sebastiani, Boston University
Matthias Seeger, University of California, Berkeley
Prakash Shenoy, University of Kansas
Yoram Singer, Hebrew University of Jerusalem
Jim Smith, University of Warwick
Alex Smola, Australian National University
Bo Thiesson, Microsoft Research
Henry Tirri, University of Helsinki
Volker Tresp, Siemens Research
Linda van der Gaag, University of Utrecht
Vladimir Vovk, Royal Holloway University of London
Martin Wainwright, University of California, Berkeley
Max Welling, University of California, Irvine
Jason Weston, NEC Labs America
Joe Whittaker, Lancaster University
Yee Whye Teh, University of California, Berkeley
Chris Williams, University of Edinburgh
John Winn, Microsoft Research
Eric Xing, Carnegie Mellon University
Xiaojin Zhu, Carnegie Mellon University

A Uniform Convergence Bound for the Area Under the ROC Curve

Shivani Agarwal, Sariel Har-Peled and Dan Roth
Department of Computer Science
University of Illinois at Urbana-Champaign
201 N. Goodwin Avenue
Urbana, IL 61801, USA
{sagarwal,sariel,danr}@cs.uiuc.edu

Abstract

The area under the ROC curve (AUC) has been advocated as an evaluation criterion for the bipartite ranking problem. We study uniform convergence properties of the AUC; in particular, we derive a distribution-free uniform convergence bound for the AUC which serves to bound the expected accuracy of a learned ranking function in terms of its empirical AUC on the training sequence from which it is learned. Our bound is expressed in terms of a new set of combinatorial parameters that we term the *bipartite rank-shatter coefficients*; these play the same role in our result as do the standard VC-dimension related shatter coefficients (also known as the growth function) in uniform convergence results for the classification error rate. A comparison of our result with a recent uniform convergence result derived by Freund et al. [9] for a quantity closely related to the AUC shows that the bound provided by our result can be considerably tighter.

1 INTRODUCTION

In many learning problems, the goal is not simply to classify objects into one of a fixed number of classes; instead, a *ranking* of objects is desired. This is the case, for example, in information retrieval problems, where one is interested in retrieving documents from some database that are 'relevant' to a given query or topic. In such problems, one wants to return to the user a list of documents that contains relevant documents at the top and irrelevant documents at the bottom; in other words, one wants a ranking of the documents such that relevant documents are ranked higher than irrelevant documents.

The problem of ranking has been studied from a learning perspective under a variety of settings [4, 10, 6, 9]. Here we consider the setting in which objects come from two categories, positive and negative; the learner is given examples of objects labeled as positive or negative, and the goal is to learn a ranking in which positive objects are ranked higher than negative ones. This captures, for example, the information retrieval problem described above; in this case, training examples consist of documents labeled as relevant (positive) or irrelevant (negative). This form of ranking problem corresponds to the 'bipartite feedback' case of [9]; we therefore refer to it as the *bipartite* ranking problem.

Formally, the setting of the bipartite ranking problem is similar to that of the binary classification problem. In both problems, there is an instance space $\mathcal{X}$ and a set of two class labels $\mathcal{Y} = \{-1, +1\}$. One is given a finite sequence of labeled training examples $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_M, y_M)) \in (\mathcal{X} \times \mathcal{Y})^M$, and the goal is to learn a function based on this training sequence. However, the form of the function to be learned in the two problems is different. In classification, one seeks a binary-valued function $h : \mathcal{X} \to \mathcal{Y}$ that predicts the class of a new instance in $\mathcal{X}$. On the other hand, in ranking, one seeks a *real-valued* function $f : \mathcal{X} \to \mathbb{R}$ that induces a ranking over $\mathcal{X}$; an instance that is assigned a higher value by f is ranked higher than one that is assigned a lower value by f.

The *area under the ROC curve* (AUC) has recently gained attention as an evaluation criterion for the bipartite ranking problem [5]. Given a ranking function $f : \mathcal{X} \to \mathbb{R}$ and a data sequence $T = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)) \in (\mathcal{X} \times \mathcal{Y})^N$ containing m positive and n negative examples, the AUC of f with respect to T, denoted $\hat{A}(f; T)$, can be expressed as the following Wilcoxon-Mann-Whitney statistic [5]:

$$\hat{A}(f;T) = \frac{1}{mn} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} \left(\mathbf{I}_{\{f(\mathbf{x}_i) > f(\mathbf{x}_j)\}} + \frac{1}{2} \mathbf{I}_{\{f(\mathbf{x}_i) = f(\mathbf{x}_j)\}} \right), \quad (1)$$

where $\mathbf{I}_{\{\cdot\}}$ denotes the indicator variable whose value is one if its argument is true and zero otherwise. The AUC of f with respect to T is thus simply the fraction of positive-negative pairs in T that are ranked correctly by f, assuming that ties are broken uniformly at random.[1]

The AUC is an empirical quantity that evaluates a ranking function with respect to a particular data sequence. What

[1] In [5], a slightly simpler form of the Wilcoxon-Mann-Whitney statistic is used, which does not account for ties.

does the empirical AUC tell us about the expected performance of a ranking function on future examples? This is the question we consider. The question has two parts, both of which are important for machine learning practice. First, what can be said about the expected performance of a ranking function based on its empirical AUC on an independent test sequence? Second, what can be said about the expected performance of a learned ranking function based on its empirical AUC on the training sequence from which it is learned? The first question is addressed in [1]; we address the second question in this paper.

We start by defining the expected ranking accuracy of a ranking function (analogous to the expected error rate of a classification function) in Section 2. Section 3 contains our uniform convergence result, which serves to bound the expected accuracy of a learned ranking function in terms of its empirical AUC on a training sequence. Our uniform convergence bound is expressed in terms of a new set of combinatorial parameters that we term the bipartite rank-shatter coefficients; these play the same role in our result as do the standard shatter coefficients (also known as the growth function) in uniform convergence results for the classification error rate. Properties of the bipartite rank-shatter coefficients are discussed in Section 4. Section 5 compares our result with a recent uniform convergence result derived by Freund et al. [9] for a quantity closely related to the AUC. We conclude with some open questions in Section 6.

2 EXPECTED RANKING ACCURACY

We begin by introducing some notation. As in classification, we shall assume that all examples are drawn randomly and independently according to some (unknown) underlying distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. The notation $\mathcal{D}_{+1}$ and $\mathcal{D}_{-1}$ will be used to denote the class-conditional distributions $\mathcal{D}_{X|Y=+1}$ and $\mathcal{D}_{X|Y=-1}$, respectively. We use an underline to denote a sequence, *e.g.*, $\underline{y} \in \mathcal{Y}^N$ to denote a sequence of elements in $\mathcal{Y}$. We shall find it convenient to decompose a data sequence $T = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)) \in (\mathcal{X} \times \mathcal{Y})^N$ into two components, $T_X = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \in \mathcal{X}^N$ and $T_Y = (y_1, \ldots, y_N) \in \mathcal{Y}^N$. Several of our results will involve the conditional distribution $\mathcal{D}_{T_X|T_Y=\underline{y}}$ for some label sequence $\underline{y} = (y_1, \ldots, y_N) \in \mathcal{Y}^N$; this distribution is simply $\mathcal{D}_{y_1} \times \ldots \times \mathcal{D}_{y_N}$.[2] As a final note of convention, we use $T \in (\mathcal{X} \times \mathcal{Y})^N$ to denote a general data sequence (*e.g.*, an independent test sequence), and $S \in (\mathcal{X} \times \mathcal{Y})^M$ to denote a training sequence.

[2]Note that, since the AUC of a ranking function f with respect to a data sequence $T \in (\mathcal{X} \times \mathcal{Y})^N$ is independent of the actual ordering of examples in the sequence, our results involving the conditional distribution $\mathcal{D}_{T_X|T_Y=\underline{y}}$ for some label sequence $\underline{y} = (y_1, \ldots, y_N) \in \mathcal{Y}^N$ depend only on the number m of positive labels in $\underline{y}$ and the number n of negative labels in $\underline{y}$. We choose to state our results in terms of the distribution $\mathcal{D}_{T_X|T_Y=\underline{y}} \equiv \mathcal{D}_{y_1} \times \ldots \times \mathcal{D}_{y_N}$ only because this is more general than $\mathcal{D}_{+1}^m \times \mathcal{D}_{-1}^n$.

Definition 1 (Expected ranking accuracy). *Let $f : \mathcal{X} \to \mathbb{R}$ be a ranking function on $\mathcal{X}$. Define the* expected ranking accuracy *(or simply* ranking accuracy*) of f, denoted by $A(f)$, as follows:*

$$A(f) = \mathbf{E}_{X \sim \mathcal{D}_{+1}, X' \sim \mathcal{D}_{-1}} \left\{ \mathbf{I}_{\{f(X) > f(X')\}} + \frac{1}{2} \mathbf{I}_{\{f(X) = f(X')\}} \right\} .$$

The ranking accuracy $A(f)$ defined above is simply the probability that an instance drawn randomly according to $\mathcal{D}_{+1}$ will be ranked higher by f than an instance drawn randomly according to $\mathcal{D}_{-1}$, assuming that ties are broken uniformly at random. The following simple lemma shows that the empirical AUC of a ranking function f is an unbiased estimator of the expected ranking accuracy of f:

Lemma 1. *Let $f : \mathcal{X} \to \mathbb{R}$ be a ranking function on $\mathcal{X}$, and let $\underline{y} = (y_1, \ldots, y_N) \in \mathcal{Y}^N$ be a finite label sequence. Then*

$$\mathbf{E}_{T_X|T_Y=\underline{y}} \left\{ \hat{A}(f; T) \right\} \quad = \quad A(f) .$$

Proof. Let m be the number of positive labels in $\underline{y}$, and n the number of negative labels in $\underline{y}$. Then from the definition of empirical AUC (Eq. (1)) and linearity of expectation, we have

$$\begin{aligned}
&\mathbf{E}_{T_X|T_Y=\underline{y}} \left\{ \hat{A}(f; T) \right\} \\
&= \frac{1}{mn} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} \mathbf{E}_{X_i \sim \mathcal{D}_{+1}, X_j \sim \mathcal{D}_{-1}} \left\{ \mathbf{I}_{\{f(X_i) > f(X_j)\}} + \frac{1}{2} \mathbf{I}_{\{f(X_i) = f(X_j)\}} \right\} \\
&= \frac{1}{mn} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} A(f) \\
&= A(f) .
\end{aligned}$$

□

3 UNIFORM CONVERGENCE BOUND

We are interested in bounding the probability that the empirical AUC of a learned ranking function f_S with respect to the (random) training sequence S from which it is learned will have a large deviation from its expected ranking accuracy, when the function f_S is chosen from a possibly infinite function class $\mathcal{F}$. The standard approach for obtaining such bounds is via uniform convergence results. In particular, we have for any $\epsilon > 0$,

$$\begin{aligned}
&\mathbf{P} \left\{ \left| \hat{A}(f_S; S) - A(f_S) \right| \geq \epsilon \right\} \\
&\qquad \leq \quad \mathbf{P} \left\{ \sup_{f \in \mathcal{F}} \left| \hat{A}(f; S) - A(f) \right| \geq \epsilon \right\} .
\end{aligned}$$

Therefore, to bound probabilities of the form on the left hand side above, it is sufficient to derive a uniform convergence result that bounds probabilities of the form on the right hand side. Our uniform convergence result for the AUC is expressed in terms of a new set of combinatorial parameters, termed the *bipartite rank-shatter coefficients*, that we define below.

Definition 2 (Bipartite rank matrix). *Let $f : \mathcal{X} \to \mathbb{R}$ be a ranking function on $\mathcal{X}$, let $m, n \in \mathbb{N}$, and let $\underline{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_m) \in \mathcal{X}^m$, $\underline{\mathbf{x}}' = (\mathbf{x}'_1, \ldots, \mathbf{x}'_n) \in \mathcal{X}^n$. Define the* bipartite rank matrix *of f with respect to $\underline{\mathbf{x}}, \underline{\mathbf{x}}'$, denoted by $\mathbf{B}_f(\underline{\mathbf{x}}, \underline{\mathbf{x}}')$, to be the matrix in $\{0, \frac{1}{2}, 1\}^{m \times n}$ whose (i, j)-th element is given by*

$$\left[\mathbf{B}_f(\underline{\mathbf{x}}, \underline{\mathbf{x}}')\right]_{ij} \quad = \quad \mathbf{I}_{\{f(\mathbf{x}_i) > f(\mathbf{x}'_j)\}} + \frac{1}{2}\mathbf{I}_{\{f(\mathbf{x}_i) = f(\mathbf{x}'_j)\}}$$

for all $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$.

Definition 3 (Bipartite rank-shatter coefficient). *Let $\mathcal{F}$ be a class of real-valued functions on $\mathcal{X}$, and let $m, n \in \mathbb{N}$. Define the (m, n)-*th bipartite rank-shatter coefficient *of $\mathcal{F}$, denoted by $r(\mathcal{F}, m, n)$, as follows:*

$$r(\mathcal{F}, m, n) \quad = \quad \max_{\underline{\mathbf{x}} \in \mathcal{X}^m, \underline{\mathbf{x}}' \in \mathcal{X}^n} \left|\left\{\mathbf{B}_f(\underline{\mathbf{x}}, \underline{\mathbf{x}}') \mid f \in \mathcal{F}\right\}\right| .$$

Clearly, for finite $\mathcal{F}$, we have $r(\mathcal{F}, m, n) \leq |\mathcal{F}|$ for all m, n. In general, $r(\mathcal{F}, m, n) \leq 3^{mn}$ for all m, n. In fact, not all 3^{mn} matrices in $\{0, \frac{1}{2}, 1\}^{m \times n}$ can be realized as bipartite rank matrices. Therefore, we have

$$r(\mathcal{F}, m, n) \quad \leq \quad \psi(m, n) ,$$

where $\psi(m, n)$ is the number of matrices in $\{0, \frac{1}{2}, 1\}^{m \times n}$ that can be realized as a bipartite rank matrix. The number $\psi(m, n)$ can be characterized in the following ways (proof omitted due to lack of space):

Theorem 1. *Let $\psi(m, n)$ be the number of matrices in $\{0, \frac{1}{2}, 1\}^{m \times n}$ that can be realized as a bipartite rank matrix $\mathbf{B}_f(\underline{\mathbf{x}}, \underline{\mathbf{x}}')$ for some $f : \mathcal{X} \to \mathbb{R}$, $\underline{\mathbf{x}} \in \mathcal{X}^m$, $\underline{\mathbf{x}}' \in \mathcal{X}^n$. Then*

1. *$\psi(m, n)$ is equal to the number of complete mixed acyclic (m, n)-bipartite graphs (where a mixed graph is one which may contain both directed and undirected edges, and where we define a cycle in such a graph as a cycle that contains at least one directed edge and in which all directed edges have the same directionality along the cycle).*
2. *$\psi(m, n)$ is equal to the number of matrices in $\{0, \frac{1}{2}, 1\}^{m \times n}$ that do not contain a sub-matrix of any of the forms shown in Table 1.*

We discuss further properties of the bipartite rank-shatter coefficients in Section 4; we first present below our uniform convergence result in terms of these coefficients. The following can be viewed as the main result of this paper. We note that our results are all distribution-free, in the sense that they hold for any distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$.

Table 1: Sub-matrices that cannot appear in a bipartite rank matrix.

$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$	$\begin{bmatrix}½ & 0\\ 0 & 1\end{bmatrix}$	$\begin{bmatrix}1 & ½\\ 0 & 1\end{bmatrix}$	$\begin{bmatrix}1 & 0\\ 0 & ½\end{bmatrix}$	$\begin{bmatrix}1 & 0\\ ½ & 1\end{bmatrix}$	$\begin{bmatrix}½ & ½\\ 0 & 1\end{bmatrix}$	$\begin{bmatrix}½ & 0\\ 0 & ½\end{bmatrix}$	$\begin{bmatrix}½ & 0\\ ½ & 1\end{bmatrix}$	$\begin{bmatrix}1 & ½\\ 0 & ½\end{bmatrix}$	$\begin{bmatrix}1 & ½\\ ½ & 1\end{bmatrix}$
$\begin{bmatrix}1 & 0\\ ½ & ½\end{bmatrix}$	$\begin{bmatrix}1 & ½\\ ½ & ½\end{bmatrix}$	$\begin{bmatrix}½ & 0\\ ½ & ½\end{bmatrix}$	$\begin{bmatrix}½ & ½\\ ½ & 1\end{bmatrix}$	$\begin{bmatrix}½ & ½\\ 0 & ½\end{bmatrix}$	$\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$	$\begin{bmatrix}½ & 1\\ 1 & 0\end{bmatrix}$	$\begin{bmatrix}0 & ½\\ 1 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 1\\ 1 & ½\end{bmatrix}$	$\begin{bmatrix}0 & 1\\ ½ & 0\end{bmatrix}$
$\begin{bmatrix}½ & ½\\ 1 & 0\end{bmatrix}$	$\begin{bmatrix}½ & 1\\ 1 & ½\end{bmatrix}$	$\begin{bmatrix}½ & 1\\ ½ & 0\end{bmatrix}$	$\begin{bmatrix}0 & ½\\ 1 & ½\end{bmatrix}$	$\begin{bmatrix}0 & ½\\ ½ & 0\end{bmatrix}$	$\begin{bmatrix}0 & 1\\ ½ & ½\end{bmatrix}$	$\begin{bmatrix}0 & ½\\ ½ & ½\end{bmatrix}$	$\begin{bmatrix}½ & 1\\ ½ & ½\end{bmatrix}$	$\begin{bmatrix}½ & ½\\ ½ & 0\end{bmatrix}$	$\begin{bmatrix}½ & ½\\ 1 & ½\end{bmatrix}$

Theorem 2. *Let $\mathcal{F}$ be a class of real-valued functions on $\mathcal{X}$, and let $\underline{y} = (y_1, \ldots, y_M) \in \mathcal{Y}^M$ be any label sequence of length $M \in \mathbb{N}$. Let m be the number of positive labels in $\underline{y}$, and $n = M - m$ the number of negative labels in $\underline{y}$. Then for any $\epsilon > 0$,*

$$\mathbf{P}_{S_X | S_Y = \underline{y}} \left\{ \sup_{f \in \mathcal{F}} \left| \hat{A}(f; S) - A(f) \right| \geq \epsilon \right\} \quad \leq \quad 4 \cdot r(\mathcal{F}, 2m, 2n) \cdot e^{-mn\epsilon^2/8(m+n)} .$$

The proof is adapted from uniform convergence proofs for the classification error rate (see, for example, [2, 8]). The main difference is that since the AUC cannot be expressed as a sum of independent random variables, more powerful inequalities are required. In particular, a result of Devroye [7] is required to bound the variance of the AUC that appears after an application of Chebyshev's inequality, and McDiarmid's inequality [12] is required in the final step of the proof where Hoeffding's inequality sufficed in the case of classification. Details are given in Appendix A.

We note that the result of Theorem 2 can be strengthened so that the conditioning is only on the numbers m and n of positive and negative labels, and not on the specific label vector $\underline{y}$.[3] From Theorem 2, we can derive a confidence interval interpretation of the bound as follows:

Corollary 1. *Let $\mathcal{F}$ be a class of real-valued functions on $\mathcal{X}$, and let $\underline{y} = (y_1, \ldots, y_M) \in \mathcal{Y}^M$ be any label sequence of length $M \in \mathbb{N}$. Let m be the number of positive labels in $\underline{y}$, and $n = M - m$ the number of negative labels in $\underline{y}$. Then for any $0 < \delta \leq 1$,*

$$\mathbf{P}_{S_X | S_Y = \underline{y}} \left\{ \sup_{f \in \mathcal{F}} \left| \hat{A}(f; S) - A(f) \right| \geq \sqrt{\frac{8(m+n)\left(\ln r(\mathcal{F}, 2m, 2n) + \ln\left(\frac{4}{\delta}\right)\right)}{mn}} \right\} \leq \delta .$$

Proof. This follows directly from Theorem 2 by setting $4 \cdot r(\mathcal{F}, 2m, 2n) \cdot e^{-mn\epsilon^2/8(m+n)} = \delta$ and solving for ϵ. □

As in the case of the large deviation bound of [1], the confidence interval above can be generalized to remove the conditioning on the label vector completely (we note that Theorem 2 cannot be generalized in this manner):

Theorem 3. *Let $\mathcal{F}$ be a class of real-valued functions on $\mathcal{X}$, and let $M \in \mathbb{N}$. Then for any $0 < \delta \leq 1$,*

$$\mathbf{P}_{S \sim \mathcal{D}^M} \left\{ \sup_{f \in \mathcal{F}} \left| \hat{A}(f; S) - A(f) \right| \geq \sqrt{\frac{8\left(\ln r\left(\mathcal{F},\, 2\rho(S_Y)M,\, 2(1-\rho(S_Y))M\right) + \ln\left(\frac{4}{\delta}\right)\right)}{\rho(S_Y)(1-\rho(S_Y))M}} \right\} \leq \delta ,$$

where $\rho(S_Y)$ denotes the proportion of positive labels in S_Y.

[3] Our thanks to an anonymous reviewer for pointing this out.

4 PROPERTIES OF BIPARTITE RANK-SHATTER COEFFICIENTS

As discussed above, we have $r(\mathcal{F},m,n) \leq \psi(m,n)$, where $\psi(m,n)$ is the number of matrices in $\{0,\frac{1}{2},1\}^{m\times n}$ that can be realized as a bipartite rank matrix. The number $\psi(m,n)$ is strictly smaller than 3^{mn}, but is still very large; in particular, $\psi(m,n) \geq 3^{\max(m,n)}$. (To see this, note that choosing any column vector in $\{0,\frac{1}{2},1\}^m$ and replicating it along the n columns or choosing any row vector in $\{0,\frac{1}{2},1\}^n$ and replicating it along the m rows results in a matrix that does not contain a sub-matrix of any of the forms shown in Table 1. The conclusion then follows from Theorem 1 (Part 2).) For the bound of Theorem 2 to be meaningful, one needs an upper bound on $r(\mathcal{F},m,n)$ that is at least slightly smaller than $e^{mn/8(m+n)}$. Below we provide one method for deriving upper bounds on $r(\mathcal{F},m,n)$; taking $\mathcal{Y}^* = \{-1,0,+1\}$, we extend slightly the standard shatter coefficients studied in classification to $\mathcal{Y}^*$-valued function classes, and then derive an upper bound on the bipartite rank-shatter coefficients $r(\mathcal{F},m,n)$ of a class of ranking functions $\mathcal{F}$ in terms of the shatter coefficients of a class of $\mathcal{Y}^*$-valued functions derived from $\mathcal{F}$.

Definition 4 (Shatter coefficient). *Let $\mathcal{Y}^* = \{-1,0,+1\}$, and let $\mathcal{H}$ be a class of $\mathcal{Y}^*$-valued functions on $\mathcal{X}$. Let $N \in \mathbb{N}$. Define the N-th* shatter coefficient *of $\mathcal{H}$, denoted by $s(\mathcal{H},N)$, as follows:*

$$s(\mathcal{H},N) \;=\; \max_{\underline{\mathbf{x}}\in\mathcal{X}^N} \left|\left\{ (h(\mathbf{x}_1),\ldots,h(\mathbf{x}_N)) \mid h\in\mathcal{H}\right\}\right|.$$

Clearly, $s(\mathcal{H},N) \leq 3^N$ for all N. Next we define a series of $\mathcal{Y}^*$-valued function classes derived from a given ranking function class. Only the second function class is used in this section; the other two are needed in Section 5. Note that we take

$$\text{sign}(u) = \begin{cases} +1 & \text{if } u>0 \\ 0 & \text{if } u=0 \\ -1 & \text{if } u<0\,. \end{cases}$$

Definition 5 (Function classes). *Let $\mathcal{F}$ be a class of real-valued functions on $\mathcal{X}$. Define the following classes of $\mathcal{Y}^*$-valued functions derived from $\mathcal{F}$:*

$$1.\quad \bar{\mathcal{F}} \;=\; \Big\{\bar{f}:\mathcal{X}\to\mathcal{Y}^* \mid \bar{f}(\mathbf{x}) = \text{sign}(f(\mathbf{x})) \text{ for some } f\in\mathcal{F}\Big\} \tag{2}$$

$$2.\quad \tilde{\mathcal{F}} \;=\; \Big\{\tilde{f}:\mathcal{X}\times\mathcal{X}\to\mathcal{Y}^* \mid \tilde{f}(\mathbf{x},\mathbf{x}') = \text{sign}(f(\mathbf{x})-f(\mathbf{x}')) \text{ for some } f\in\mathcal{F}\Big\} \tag{3}$$

$$3.\quad \check{\mathcal{F}} \;=\; \Big\{\check{f}_{\mathbf{z}}:\mathcal{X}\to\mathcal{Y}^* \mid \check{f}_{\mathbf{z}}(\mathbf{x}) = \text{sign}(f(\mathbf{x})-f(\mathbf{z})) \text{ for some } f\in\mathcal{F},\mathbf{z}\in\mathcal{X}\Big\} \tag{4}$$

Theorem 4. *Let $\mathcal{F}$ be a class of real-valued functions on $\mathcal{X}$, and let $\tilde{\mathcal{F}}$ be the class of $\mathcal{Y}^*$-valued functions on $\mathcal{X}\times\mathcal{X}$ defined by Eq. (3). Then for all $m,n\in\mathbb{N}$,*

$$r(\mathcal{F},m,n) \;\leq\; s(\tilde{\mathcal{F}},mn)\,.$$

Proof. For any $m,n\in\mathbb{N}$, we have[4]

$$\begin{aligned}
&r(\mathcal{F},m,n)\\
&= \max_{\underline{\mathbf{x}}\in\mathcal{X}^m,\underline{\mathbf{x}}'\in\mathcal{X}^n} \left|\left\{ \left[\mathbf{I}_{\{f(\mathbf{x}_i)>f(\mathbf{x}'_j)\}} + \frac{1}{2}\mathbf{I}_{\{f(\mathbf{x}_i)=f(\mathbf{x}'_j)\}}\right] \;\middle|\; f\in\mathcal{F}\right\}\right|\\
&= \max_{\underline{\mathbf{x}}\in\mathcal{X}^m,\underline{\mathbf{x}}'\in\mathcal{X}^n} \left|\left\{ \left[\mathbf{I}_{\{\tilde{f}(\mathbf{x}_i,\mathbf{x}'_j)=+1\}} + \frac{1}{2}\mathbf{I}_{\{\tilde{f}(\mathbf{x}_i,\mathbf{x}'_j)=0\}}\right] \;\middle|\; \tilde{f}\in\tilde{\mathcal{F}}\right\}\right|\\
&= \max_{\underline{\mathbf{x}}\in\mathcal{X}^m,\underline{\mathbf{x}}'\in\mathcal{X}^n} \left|\left\{ \left[\tilde{f}(\mathbf{x}_i,\mathbf{x}'_j)\right] \;\middle|\; \tilde{f}\in\tilde{\mathcal{F}}\right\}\right|\\
&\leq \max_{\mathbf{x},\mathbf{x}'\in\mathcal{X}^{m\times n}} \left|\left\{ \left[\tilde{f}(\mathbf{x}_{ij},\mathbf{x}'_{ij})\right] \;\middle|\; \tilde{f}\in\tilde{\mathcal{F}}\right\}\right|\\
&= \max_{\underline{\mathbf{x}},\underline{\mathbf{x}}'\in\mathcal{X}^{mn}} \left|\left\{ \left(\tilde{f}(\mathbf{x}_1,\mathbf{x}'_1),\ldots,\tilde{f}(\mathbf{x}_{mn},\mathbf{x}'_{mn})\right) \;\middle|\; \tilde{f}\in\tilde{\mathcal{F}}\right\}\right|\\
&= s(\tilde{\mathcal{F}},mn)\,. \qquad \square
\end{aligned}$$

Below we make use of the above result to derive a polynomial upper bound on the bipartite rank-shatter coefficients for the case of linear ranking functions. We note that the same method can be used to establish similar upper bounds for higher-order polynomial ranking functions and other algebraically well-behaved function classes.

Lemma 2. *For $d\in\mathbb{N}$, let $\mathcal{F}_{\text{lin}(d)}$ denote the class of linear ranking functions on $\mathbb{R}^d$:*

$$\mathcal{F}_{\text{lin}(d)} \;=\; \Big\{f:\mathbb{R}^d\to\mathbb{R} \mid f(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x}+b \text{ for some } \mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}\Big\}.$$

Then for all $N\in\mathbb{N}$,

$$s(\tilde{\mathcal{F}}_{\text{lin}(d)},N) \;\leq\; (2eN/d)^d.$$

Proof. We have,

$$\tilde{\mathcal{F}}_{\text{lin}(d)} \;=\; \Big\{\tilde{f}:\mathbb{R}^d\times\mathbb{R}^d\to\mathcal{Y}^* \mid \tilde{f}(\mathbf{x},\mathbf{x}') = \text{sign}(\mathbf{w}\cdot(\mathbf{x}-\mathbf{x}')) \text{ for some } \mathbf{w}\in\mathbb{R}^d\Big\}.$$

Let $(\mathbf{x}_1,\mathbf{x}'_1),\ldots,(\mathbf{x}_N,\mathbf{x}'_N)$ be any N points in $\mathbb{R}^d\times\mathbb{R}^d$, and consider the 'dual' weight space corresponding to $\mathbf{w}\in\mathbb{R}^d$. Each point $(\mathbf{x}_i,\mathbf{x}'_i)$ defines a hyperplane $(\mathbf{x}_i-\mathbf{x}'_i)$ in this space; the N points thus give rise to an arrangement of N hyperplanes in $\mathbb{R}^d$. It is easily seen that the number of sign patterns $(\tilde{f}(\mathbf{x}_1,\mathbf{x}'_1),\ldots,\tilde{f}(\mathbf{x}_N,\mathbf{x}'_N))$ that can be realized by functions $\tilde{f}\in\tilde{\mathcal{F}}$ is equal to the total number of faces of this arrangement [11], which is at most [3]

$$\sum_{k=0}^{d}\sum_{i=d-k}^{d}\binom{i}{d-k}\binom{N}{i} \;=\; \sum_{i=0}^{d}2^i\binom{N}{i} \;\leq\; (2eN/d)^d\,.$$

Since the N points were arbitrary, the result follows. $\square$

Theorem 5. *For $d\in\mathbb{N}$, let $\mathcal{F}_{\text{lin}(d)}$ denote the class of linear ranking functions on $\mathbb{R}^d$ (defined in Lemma 2 above). Then for all $m,n\in\mathbb{N}$,*

$$r(\mathcal{F}_{\text{lin}(d)},m,n) \;\leq\; (2emn/d)^d.$$

Proof. This follows immediately from Theorem 4 and Lemma 2. $\square$

[4] We use the notation $[a_{ij}]$ to denote a matrix whose $(i,j)^{\text{th}}$ element is a_{ij}. The dimensions of such a matrix should be clear from context.

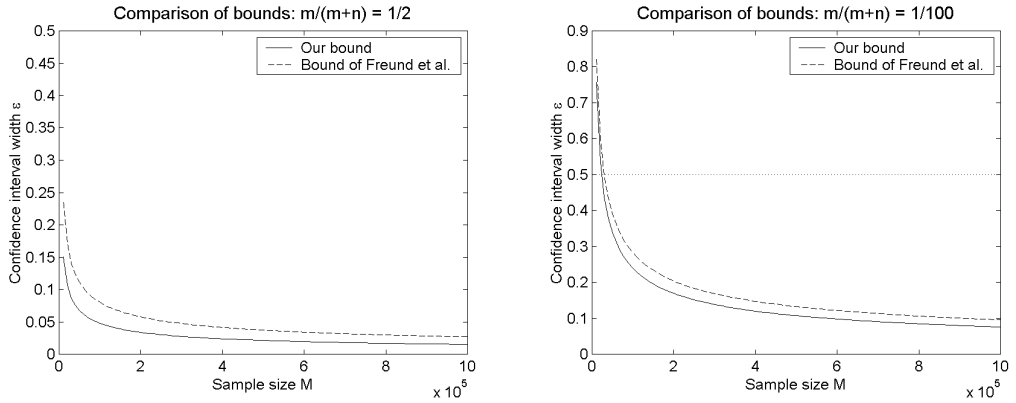


Figure 1: A comparison of our uniform convergence bound with that of [9] for the class of linear ranking functions on $\mathbb{R}$. The plots are for $\delta = 0.01$ and show how the confidence interval width ϵ given by the two bounds varies with the sample size M, for various values of $m/(m+n)$. In all cases where the bounds are meaningful ($\epsilon < 0.5$), our bound is tighter.

5 COMPARISON WITH BOUND OF FREUND ET AL.

Freund et al. [9] recently derived a uniform convergence bound for a quantity closely related to the AUC, namely the ranking loss for the bipartite ranking problem. As pointed out in [5], the bipartite ranking loss is equal to one minus the AUC; the uniform convergence bound of [9] therefore implies a uniform convergence bound for the AUC.[5] Although the result in [9] is given only for function classes considered by their RankBoost algorithm, their technique is generally applicable. We state their result below, using our notation, for the general case (*i.e.*, function classes not restricted to those considered by RankBoost), and then offer a comparison of our bound with theirs. As in [9], the result is given in the form of a confidence interval.[6]

Theorem 6 (Generalization of [9], Theorem 3). *Let $\mathcal{F}$ be a class of real-valued functions on $\mathcal{X}$, and let $\underline{y} = (y_1, \ldots, y_M) \in \mathcal{Y}^M$ be any label sequence of length $M \in \mathbb{N}$. Let m be the number of positive labels in $\underline{y}$, and $n = M - m$ the number of negative labels in $\underline{y}$. Then for any $0 < \delta \leq 1$,*

$$\mathbf{P}_{S_X|S_Y=\underline{y}}\left\{ \sup_{f\in\mathcal{F}} \left|\hat{A}(f;S) - A(f)\right| \geq 2\sqrt{\frac{\ln s(\check{\mathcal{F}},2m) + \ln\left(\frac{12}{\delta}\right)}{m}} + 2\sqrt{\frac{\ln s(\check{\mathcal{F}},2n) + \ln\left(\frac{12}{\delta}\right)}{n}} \right\} \leq \delta\,,$$

where $\check{\mathcal{F}}$ is the class of $\mathcal{Y}^*$-valued functions on $\mathcal{X}$ defined by Eq. (4).

[5] As in the AUC definition of [5], the ranking loss defined in [9] does not account for ties; this is easily remedied.

[6] The result in [9] was stated in terms of the VC dimension, but the basic result can be stated in terms of shatter coefficients. Due to our AUC definition which accounts for ties, the standard shatter coefficients are replaced here with the extended shatter coefficients defined above for $\mathcal{Y}^*$-valued function classes.

The proof follows that of [9] and is omitted. We now compare the uniform convergence bound derived in Section 3 with that of Freund et al. for a simple function class for which the quantities involved in both bounds (namely, $r(\mathcal{F},2m,2n)$ and $s(\check{\mathcal{F}},2m), s(\check{\mathcal{F}},2n)$) can be characterized exactly. Specifically, consider the function class $\mathcal{F}_{\text{lin}(1)}$ of linear ranking functions on $\mathbb{R}$, given by

$$\mathcal{F}_{\text{lin}(1)} = \{f : \mathbb{R}\to\mathbb{R} \mid f(x) = wx + b \text{ for some } w \in \mathbb{R}, b \in \mathbb{R}\}\,.$$

Although $\mathcal{F}_{\text{lin}(1)}$ is an infinite function class, it is easy to verify that $r(\mathcal{F}_{\text{lin}(1)},m,n) = 3$ for all $m,n \in \mathbb{N}$. (To see this, note that for any set of $m+n$ distinct points in $\mathbb{R}$, one can obtain exactly three different ranking behaviours with functions in $\mathcal{F}_{\text{lin}(1)}$: one by setting $w > 0$, another by setting $w < 0$, and the third by setting $w = 0$.) On the other hand, $s(\check{\mathcal{F}}_{\text{lin}(1)},N) = 4N+1$ for all $N \geq 2$, since $\check{\mathcal{F}}_{\text{lin}(1)} = \bar{\mathcal{F}}_{\text{lin}(1)}$ (see Eq. (2)) and, as is easily verified, the number of sign patterns on $N \geq 2$ distinct points in $\mathbb{R}$ that can be realized by functions in $\bar{\mathcal{F}}_{\text{lin}(1)}$ is $4N+1$. We thus get from our result (Corollary 1) that

$$\mathbf{P}_{S_X|S_Y=\underline{y}}\left\{ \sup_{f\in\mathcal{F}_{\text{lin}(1)}} \left|\hat{A}(f;S) - A(f)\right| \geq \sqrt{\frac{8(m+n)\left(\ln 3 + \ln\left(\frac{4}{\delta}\right)\right)}{mn}} \right\} \leq \delta\,,$$

and from the result of Freund et al. (Theorem 6) that

$$\mathbf{P}_{S_X|S_Y=\underline{y}}\left\{ \sup_{f\in\mathcal{F}_{\text{lin}(1)}} \left|\hat{A}(f;S) - A(f)\right| \geq 2\sqrt{\frac{\ln(8m+1) + \ln\left(\frac{12}{\delta}\right)}{m}} + 2\sqrt{\frac{\ln(8n+1) + \ln\left(\frac{12}{\delta}\right)}{n}} \right\} \leq \delta\,.$$

The above bounds are plotted in Figure 1 for $\delta = 0.01$ and various values of $m/(m+n)$. As can be seen, the bound provided by our result is considerably tighter.

6 CONCLUSION & OPEN QUESTIONS

We have derived a distribution-free uniform convergence bound for the area under the ROC curve (AUC), a quantity used as an evaluation criterion for the bipartite ranking problem. Our bound is expressed in terms of a new set of combinatorial parameters that we have termed the bipartite rank-shatter coefficients. These coefficients define a new measure of complexity for real-valued function classes and play the same role in our result as do the standard VC-dimension related shatter coefficients in uniform convergence results for the classification error rate.

For the case of linear ranking functions on $\mathbb{R}$, for which we could compute the bipartite rank-shatter coefficients exactly, we have shown that our uniform convergence bound is considerably tighter than a recent bound of Freund et al. [9], which is expressed directly in terms of standard shatter coefficients from results for classification. This suggests that the bipartite rank-shatter coefficients we have introduced may be a more appropriate complexity measure for studying the bipartite ranking problem. However, in order to take advantage of our results, one needs to be able to characterize these coefficients for the class of ranking functions of interest. The biggest open question that arises from our study is, for what other function classes $\mathcal{F}$ can the bipartite rank-shatter coefficients $r(\mathcal{F}, m, n)$ be characterized? We have derived in Theorem 4 a general upper bound on the bipartite rank-shatter coefficients of a function class $\mathcal{F}$ in terms of the standard shatter coefficients of the function class $\tilde{\mathcal{F}}$ (see Eq. (3)); this allows us to establish a polynomial upper bound on the bipartite rank-shatter coefficients for linear ranking functions on $\mathbb{R}^d$ and other algebraically well-behaved function classes. However, this upper bound is inherently loose (see proof of Theorem 4). Is it possible to find tighter upper bounds on $r(\mathcal{F}, m, n)$ than that given by Theorem 4?

Our study also raises several other interesting questions. First, can we establish analogous complexity measures and generalization bounds for other forms of ranking problems (*i.e.*, other than bipartite)? Second, do there exist data-dependent bounds for ranking, analogous to existing margin bounds for classification? Finally, it also remains an open question whether tighter generalization bounds for the AUC can be derived using different proof techniques.

Acknowledgements

We would like to thank Thore Graepel and Ralf Herbrich for discussions related to this work and for pointing out to us the graph-based interpretation of bipartite rank matrices used in Theorem 1. We are also very grateful to an anonymous reviewer, and to Thore Graepel and Ralf Herbrich again, for helping us identify an important mistake in an earlier version of our results. This research was supported in part by NSF ITR grants IIS 00-85980 and IIS 00-85836 and a grant from the ONR-TRECC program.

References

[1] Shivani Agarwal, Thore Graepel, Ralf Herbrich, and Dan Roth. A large deviation bound for the area under the ROC curve. In *Advances in Neural Information Processing Systems 17*. MIT Press, 2005. To appear.

[2] Martin Anthony and Peter Bartlett. *Learning in Neural Networks: Theoretical Foundations*. Cambridge University Press, 1999.

[3] R. C. Buck. Partition of space. *American Mathematical Monthly*, 50:2541–544, 1943.

[4] William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

[5] Corinna Cortes and Mehryar Mohri. AUC optimization vs. error rate minimization. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

[6] Koby Crammer and Yoram Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems 14*, pages 641–647. MIT Press, 2002.

[7] Luc Devroye. Exponential inequalities in nonparametric estimation. In G. Roussas, editor, *Nonparametric Functional Estimation and Related Topics*, NATO ASI Series, pages 31–44. Kluwer Academic Publishers, 1991.

[8] Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, New York, 1996.

[9] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[10] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers*, pages 115–132, 2000.

[11] Jiří Matoušek. *Lectures on Discrete Geometry*. Springer-Verlag, New York, 2002.

[12] Colin McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics 1989*, pages 148–188. Cambridge University Press, 1989.

A Proof of Theorem 2

Our proof makes use of the following two results [12, 7] that bound the probability of a large deviation and the variance, respectively, of any function of a sample for which a single change in the sample has limited effect:

Theorem 7 (McDiarmid, 1989). *Let $X_1, \ldots, X_N$ be independent random variables with X_k taking values in a set A_k for each k. Let $\phi : (A_1 \times \cdots \times A_N) \to \mathbb{R}$ be such that*

$$\sup_{x_i \in A_i, x'_k \in A_k} \Big| \phi(x_1, \ldots, x_N) - \phi(x_1, \ldots, x_{k-1}, x'_k, x_{k+1}, \ldots, x_N) \Big| \;\leq\; c_k \,.$$

Then for any $\epsilon > 0$,

$$\mathbf{P}\left\{ |\phi(X_1, \ldots, X_N) - \mathbf{E}\{\phi(X_1, \ldots, X_N)\}| \geq \epsilon \right\} \;\leq\; 2e^{-2\epsilon^2 / \sum_{k=1}^N c_k^2} \,.$$

Theorem 8 (Devroye, 1991; Devroye et al., 1996, Theorem 9.3). *Under the conditions of Theorem 7,*

$$\mathbf{Var}\left\{\phi(X_1, \ldots, X_N)\right\} \;\leq\; \frac{1}{4} \sum_{k=1}^N c_k^2 \,.$$

The following lemma establishes that a change in a single instance in a data sequence has a limited effect on the AUC of a ranking function with respect to the data sequence:

Lemma 3. *Let $f : \mathcal{X} \to \mathbb{R}$ be a ranking function on $\mathcal{X}$ and let $\underline{y} = (y_1, \ldots, y_N) \in \mathcal{Y}^N$ be a finite label sequence. Let m be the number of positive labels in $\underline{y}$ and n the number of negative labels in $\underline{y}$. Let $\phi : \mathcal{X}^N \to \mathbb{R}$ be defined as follows:*

$$\phi\left(\mathbf{x}_1, \ldots, \mathbf{x}_N\right) \;=\; \hat{A}\left(f; ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N))\right) \,.$$

Then for all $\mathbf{x}_i, \mathbf{x}'_k \in \mathcal{X}$,

$$\left|\phi(\mathbf{x}_1, \ldots, \mathbf{x}_N) - \phi(\mathbf{x}_1, \ldots, \mathbf{x}_{k-1}, \mathbf{x}'_k, \mathbf{x}_{k+1} \ldots, \mathbf{x}_N)\right| \;\leq\; c_k \,,$$

where $c_k = 1/m$ if $y_k = +1$ and $c_k = 1/n$ if $y_k = -1$.

Proof. For each k such that $y_k = +1$, we have

$$\begin{aligned}
&\left|\phi(\mathbf{x}_1, \ldots, \mathbf{x}_N) - \phi(\mathbf{x}_1, \ldots, \mathbf{x}_{k-1}, \mathbf{x}'_k, \mathbf{x}_{k+1} \ldots, \mathbf{x}_N)\right| \\
&\quad= \frac{1}{mn} \left| \sum_{\{j : y_j = -1\}} \left(\left(\mathbf{I}_{\{f(\mathbf{x}_k) > f(\mathbf{x}_j)\}} + \frac{1}{2} \mathbf{I}_{\{f(\mathbf{x}_k) = f(\mathbf{x}_j)\}} \right) - \left(\mathbf{I}_{\{f(\mathbf{x}'_k) > f(\mathbf{x}_j)\}} + \frac{1}{2} \mathbf{I}_{\{f(\mathbf{x}'_k) = f(\mathbf{x}_j)\}} \right) \right) \right| \\
&\quad\leq \frac{1}{mn} n \\
&\quad= \frac{1}{m} \,.
\end{aligned}$$

The case $y_k = -1$ can be proved similarly. □

We are now ready to give the main proof:

Proof (of Theorem 2). The proof is adapted from proofs of uniform convergence for the classification error rate given in [2, 8]. It consists of four steps.

Step 1. First symmetrization by a ghost sample.

For each $k \in \{1, \ldots, M\}$, define the random variable $\tilde{X}_k$ such that $X_k, \tilde{X}_k$ are independent and identically distributed. Let $\tilde{S}_X = (\tilde{X}_1, \ldots, \tilde{X}_M)$, and denote by $\tilde{S}$ the joint sequence $(\tilde{S}_X, \underline{y})$. Then for any $\epsilon > 0$ satisfying $mn\epsilon^2/(m+n) \geq 2$, we have

$$\begin{aligned}
&\mathbf{P}_{S_X | S_Y = \underline{y}} \left\{ \sup_{f \in \mathcal{F}} \left| \hat{A}(f; S) - A(f) \right| \geq \epsilon \right\} \\
&\qquad\leq\; 2\mathbf{P}_{S_X \tilde{S}_X | S_Y = \underline{y}} \left\{ \sup_{f \in \mathcal{F}} \left| \hat{A}(f; S) - \hat{A}(f; \tilde{S}) \right| \geq \frac{\epsilon}{2} \right\} \,.
\end{aligned}$$

To see this, let $f^*_S \in \mathcal{F}$ be a function for which $|\hat{A}(f^*_S; S) - A(f^*_S)| \geq \epsilon$ if such a function exists, and let f^*_S be a fixed function in $\mathcal{F}$ otherwise. Then

$$\begin{aligned}
&\mathbf{P}_{S_X \tilde{S}_X | S_Y = \underline{y}} \left\{ \sup_{f \in \mathcal{F}} \left| \hat{A}(f; S) - \hat{A}(f; \tilde{S}) \right| \geq \frac{\epsilon}{2} \right\} \\
&\quad\geq\; \mathbf{P}_{S_X \tilde{S}_X | S_Y = \underline{y}} \left\{ \left| \hat{A}(f^*_S; S) - \hat{A}(f^*_S; \tilde{S}) \right| \geq \frac{\epsilon}{2} \right\} \\
&\quad\geq\; \mathbf{P}_{S_X \tilde{S}_X | S_Y = \underline{y}} \left\{ \left\{ \left| \hat{A}(f^*_S; S) - A(f^*_S) \right| \geq \epsilon \right\} \cap \left\{ \left| \hat{A}(f^*_S; \tilde{S}) - A(f^*_S) \right| \leq \frac{\epsilon}{2} \right\} \right\} \\
&\quad=\; \mathbf{E}_{S_X | S_Y = \underline{y}} \left\{ \mathbf{I}_{\{|\hat{A}(f^*_S; S) - A(f^*_S)| \geq \epsilon\}} \times \mathbf{P}_{\tilde{S}_X | S_X, S_Y = \underline{y}} \left\{ \left| \hat{A}(f^*_S; \tilde{S}) - A(f^*_S) \right| \leq \frac{\epsilon}{2} \right\} \right\} \,. \qquad (5)
\end{aligned}$$

The conditional probability inside can be bounded using Chebyshev's inequality (and Lemma 1):

$$\begin{aligned}
&\mathbf{P}_{\tilde{S}_X | S_X, S_Y = \underline{y}} \left\{ \left| \hat{A}(f^*_S; \tilde{S}) - A(f^*_S) \right| \leq \frac{\epsilon}{2} \right\} \\
&\qquad\geq\; 1 - \frac{\mathbf{Var}_{\tilde{S}_X | S_X, S_Y = \underline{y}} \left\{ \hat{A}(f^*_S; \tilde{S}) \right\}}{\epsilon^2/4} \,.
\end{aligned}$$

Now, by Lemma 3 and Theorem 8, we have

$$\begin{aligned}
&\mathbf{Var}_{\tilde{S}_X | S_X, S_Y = \underline{y}} \left\{ \hat{A}(f^*_S; \tilde{S}) \right\} \\
&\qquad\leq\; \frac{1}{4} \left(m \Big(\frac{1}{m}\Big)^2 + n \Big(\frac{1}{n}\Big)^2 \right) \;=\; \frac{m+n}{4mn} \,.
\end{aligned}$$

This gives

$$\mathbf{P}_{\tilde{S}_X | S_X, S_Y = \underline{y}} \left\{ \left| \hat{A}(f^*_S; \tilde{S}) - A(f^*_S) \right| \leq \frac{\epsilon}{2} \right\} \;\geq\; 1 - \frac{m+n}{mn\epsilon^2} \;\geq\; \frac{1}{2} \,,$$

whenever $mn\epsilon^2/(m+n) \geq 2$. Thus, from Eq. (5) and the definition of f^*_S, we have

$$\begin{aligned}
&\mathbf{P}_{S_X \tilde{S}_X | S_Y = \underline{y}} \left\{ \sup_{f \in \mathcal{F}} \left| \hat{A}(f; S) - \hat{A}(f; \tilde{S}) \right| \geq \frac{\epsilon}{2} \right\} \\
&\qquad\geq\; \frac{1}{2} \mathbf{E}_{S_X | S_Y = \underline{y}} \left\{ \mathbf{I}_{\{|\hat{A}(f^*_S; S) - A(f^*_S)| \geq \epsilon\}} \right\} \\
&\qquad=\; \frac{1}{2} \mathbf{P}_{S_X | S_Y = \underline{y}} \left\{ \left| \hat{A}(f^*_S; S) - A(f^*_S) \right| \geq \epsilon \right\} \\
&\qquad\geq\; \frac{1}{2} \mathbf{P}_{S_X | S_Y = \underline{y}} \left\{ \sup_{f \in \mathcal{F}} \left| \hat{A}(f; S) - A(f) \right| \geq \epsilon \right\} \,.
\end{aligned}$$

Step 2. Second symmetrization by permutations.

Let Γ_M be the set of all permutations of $\{X_1, \ldots, X_M, \tilde{X}_1, \ldots, \tilde{X}_M\}$ that swap X_k and $\tilde{X}_k$, for all k in some subset of $\{1, \ldots, M\}$. In other words, for all $\sigma \in \Gamma_M$ and $k \in \{1, \ldots, M\}$, either $\sigma(X_k) = X_k$, in which case $\sigma(\tilde{X}_k) = \tilde{X}_k$, or $\sigma(X_k) = \tilde{X}_k$, in which case $\sigma(\tilde{X}_k) = X_k$. Denote $\sigma(S_X) = (\sigma(X_1), \ldots, \sigma(X_M))$, and $\sigma(\tilde{S}_X) = (\sigma(\tilde{X}_1), \ldots, \sigma(\tilde{X}_M))$. Now, define

$$\beta_f(S_X,\tilde{S}_X) \equiv \frac{1}{mn}\sum_{\{i:y_i=+1\}}\sum_{\{j:y_j=-1\}}\left(\left(\mathbf{I}_{\{f(X_i)>f(X_j)\}}+\frac{1}{2}\mathbf{I}_{\{f(X_i)=f(X_j)\}}\right)-\left(\mathbf{I}_{\{f(\tilde{X}_i)>f(\tilde{X}_j)\}}+\frac{1}{2}\mathbf{I}_{\{f(\tilde{X}_i)=f(\tilde{X}_j)\}}\right)\right).$$

Then clearly, since $X_k, \tilde{X}_k$ are i.i.d. for each k, for any $\sigma \in \Gamma_M$ we have that the distribution of

$$\sup_{f\in\mathcal{F}}\left|\beta_f(S_X,\tilde{S}_X)\right|$$

is the same as the distribution of

$$\sup_{f\in\mathcal{F}}\left|\beta_f(\sigma(S_X),\sigma(\tilde{S}_X))\right| .$$

Therefore, using $\mathcal{U}(D)$ to denote the uniform distribution over a discrete set D, we have the following (note that except where specified otherwise, all probabilities and expectations below are with respect to the distribution $\mathcal{D}_{S_X\tilde{S}_X|S_Y=\underline{y}}$):

$$\begin{aligned}
&\mathbf{P}\left\{\sup_{f\in\mathcal{F}}\left|\hat{A}(f;S)-\hat{A}(f;\tilde{S})\right|\geq\frac{\epsilon}{2}\right\}\\
&\quad=\mathbf{P}\left\{\sup_{f\in\mathcal{F}}\left|\beta_f(S_X,\tilde{S}_X)\right|\geq\frac{\epsilon}{2}\right\}\\
&\quad=\frac{1}{|\Gamma_M|}\sum_{\sigma\in\Gamma_M}\mathbf{P}\left\{\sup_{f\in\mathcal{F}}\left|\beta_f(\sigma(S_X),\sigma(\tilde{S}_X))\right|\geq\frac{\epsilon}{2}\right\}\\
&\quad=\frac{1}{|\Gamma_M|}\sum_{\sigma\in\Gamma_M}\mathbf{E}\left\{\mathbf{I}_{\left\{\sup_{f\in\mathcal{F}}\left|\beta_f(\sigma(S_X),\sigma(\tilde{S}_X))\right|\geq\frac{\epsilon}{2}\right\}}\right\}\\
&\quad=\mathbf{E}\left\{\frac{1}{|\Gamma_M|}\sum_{\sigma\in\Gamma_M}\mathbf{I}_{\left\{\sup_{f\in\mathcal{F}}\left|\beta_f(\sigma(S_X),\sigma(\tilde{S}_X))\right|\geq\frac{\epsilon}{2}\right\}}\right\}\\
&\quad=\mathbf{E}\left\{\mathbf{P}_{\sigma\sim\mathcal{U}(\Gamma_M)}\left\{\sup_{f\in\mathcal{F}}\left|\beta_f(\sigma(S_X),\sigma(\tilde{S}_X))\right|\geq\frac{\epsilon}{2}\right\}\right\}\\
&\quad\leq\max_{\underline{\mathbf{x}},\underline{\tilde{\mathbf{x}}}\in\mathcal{X}^M}\mathbf{P}_{\sigma\sim\mathcal{U}(\Gamma_M)}\left\{\sup_{f\in\mathcal{F}}\left|\beta_f(\sigma(\underline{\mathbf{x}}),\sigma(\underline{\tilde{\mathbf{x}}}))\right|\geq\frac{\epsilon}{2}\right\}.
\end{aligned}$$

Step 3. Reduction to a finite class.

We wish to bound the quantity on the right hand side above. From the definition of bipartite rank matrices (Definition 2), it follows that for any $\underline{\mathbf{x}},\underline{\tilde{\mathbf{x}}}\in\mathcal{X}^M$, as f ranges over $\mathcal{F}$, the number of different random variables

$$\left|\beta_f(\sigma(\underline{\mathbf{x}}),\sigma(\underline{\tilde{\mathbf{x}}}))\right|$$

is at most the number of different bipartite rank matrices $\mathbf{B}_f(\underline{\mathbf{z}},\underline{\mathbf{z}}')$ that can be realized by functions in $\mathcal{F}$, where $\underline{\mathbf{z}}\in\mathcal{X}^{2m}$ contains $\mathbf{x}_i,\tilde{\mathbf{x}}_i$ for $i:y_i=+1$ and $\underline{\mathbf{z}}'\in\mathcal{X}^{2n}$ contains $\mathbf{x}_j,\tilde{\mathbf{x}}_j$ for $j:y_j=-1$. This number, by definition, cannot exceed $r(\mathcal{F},2m,2n)$ (see the definition of bipartite rank-shatter coefficients, Definition 3). Therefore, the supremum in the above probability is a maximum of at most $r(\mathcal{F},2m,2n)$ random variables. Thus, by the union bound, we get for any $\underline{\mathbf{x}},\underline{\tilde{\mathbf{x}}}\in\mathcal{X}^M$,

$$\begin{aligned}
&\mathbf{P}_{\sigma\sim\mathcal{U}(\Gamma_M)}\left\{\sup_{f\in\mathcal{F}}\left|\beta_f(\sigma(\underline{\mathbf{x}}),\sigma(\underline{\tilde{\mathbf{x}}}))\right|\geq\frac{\epsilon}{2}\right\}\\
&\quad\leq\ r(\mathcal{F},2m,2n)\cdot\sup_{f\in\mathcal{F}}\mathbf{P}_{\sigma\sim\mathcal{U}(\Gamma_M)}\left\{\left|\beta_f(\sigma(\underline{\mathbf{x}}),\sigma(\underline{\tilde{\mathbf{x}}}))\right|\geq\frac{\epsilon}{2}\right\}.
\end{aligned}$$

Step 4. McDiarmid's inequality.

Notice that for any $\underline{\mathbf{x}},\underline{\tilde{\mathbf{x}}}\in\mathcal{X}^M$, we can write

$$\begin{aligned}
&\mathbf{P}_{\sigma\sim\mathcal{U}(\Gamma_M)}\left\{\left|\beta_f(\sigma(\underline{\mathbf{x}}),\sigma(\underline{\tilde{\mathbf{x}}}))\right|\geq\frac{\epsilon}{2}\right\}\\
&\qquad=\ \mathbf{P}_{\underline{W}\sim\mathcal{U}(\prod_{k=1}^M\{\mathbf{x}_k,\tilde{\mathbf{x}}_k\})}\left\{\left|\beta_f(\underline{W},\underline{\tilde{W}})\right|\geq\frac{\epsilon}{2}\right\},
\end{aligned}$$

where $\underline{W}=(W_1,\ldots,W_M)$, $\underline{\tilde{W}}=(\tilde{W}_1,\ldots,\tilde{W}_M)$ and

$$\tilde{W}_k=\begin{cases}\tilde{\mathbf{x}}_k, & \text{if } W_k=\mathbf{x}_k\\ \mathbf{x}_k, & \text{if } W_k=\tilde{\mathbf{x}}_k\end{cases}.$$

Now, for any $f\in\mathcal{F}$,

$$\mathbf{E}_{\underline{W}\sim\mathcal{U}(\prod_{k=1}^M\{\mathbf{x}_k,\tilde{\mathbf{x}}_k\})}\left\{\beta_f(\underline{W},\underline{\tilde{W}})\right\}\ =\ 0,$$

since for all $i:y_i=+1$ and $j:y_j=-1$,

$$\begin{aligned}
&\mathbf{E}_{W_i\sim\mathcal{U}(\{\mathbf{x}_i,\tilde{\mathbf{x}}_i\}),W_j\sim\mathcal{U}(\{\mathbf{x}_j,\tilde{\mathbf{x}}_j\})}\left\{\mathbf{I}_{\{f(W_i)>f(W_j)\}}-\mathbf{I}_{\{f(\tilde{W}_i)>f(\tilde{W}_j)\}}\right\}\\
&\quad=\frac{1}{4}\Big(\left(\mathbf{I}_{\{f(\mathbf{x}_i)>f(\mathbf{x}_j)\}}-\mathbf{I}_{\{f(\tilde{\mathbf{x}}_i)>f(\tilde{\mathbf{x}}_j)\}}\right)+\\
&\qquad\left(\mathbf{I}_{\{f(\tilde{\mathbf{x}}_i)>f(\mathbf{x}_j)\}}-\mathbf{I}_{\{f(\mathbf{x}_i)>f(\tilde{\mathbf{x}}_j)\}}\right)+\\
&\qquad\left(\mathbf{I}_{\{f(\mathbf{x}_i)>f(\tilde{\mathbf{x}}_j)\}}-\mathbf{I}_{\{f(\tilde{\mathbf{x}}_i)>f(\mathbf{x}_j)\}}\right)+\\
&\qquad\left(\mathbf{I}_{\{f(\tilde{\mathbf{x}}_i)>f(\tilde{\mathbf{x}}_j)\}}-\mathbf{I}_{\{f(\mathbf{x}_i)>f(\mathbf{x}_j)\}}\right)\Big)\\
&\quad=0,
\end{aligned}$$

and similarly,

$$\begin{aligned}
&\mathbf{E}_{W_i\sim\mathcal{U}(\{\mathbf{x}_i,\tilde{\mathbf{x}}_i\}),W_j\sim\mathcal{U}(\{\mathbf{x}_j,\tilde{\mathbf{x}}_j\})}\left\{\mathbf{I}_{\{f(W_i)=f(W_j)\}}-\mathbf{I}_{\{f(\tilde{W}_i)=f(\tilde{W}_j)\}}\right\}\\
&\quad=0.
\end{aligned}$$

Also, it can be verified that for any $f\in\mathcal{F}$, a change in the value of a single random variable W_k can bring about a change of at most $2/m$ in the value of

$$\beta_f(\underline{W},\underline{\tilde{W}})$$

for $k:y_k=+1$, and a change of at most $2/n$ for $k:y_k=-1$. Therefore, by McDiarmid's inequality (Theorem 7), it follows that for any $f\in\mathcal{F}$,

$$\begin{aligned}
&\mathbf{P}_{\underline{W}\sim\mathcal{U}(\prod_{k=1}^M\{\mathbf{x}_k,\tilde{\mathbf{x}}_k\})}\left\{\left|\beta_f(\underline{W},\underline{\tilde{W}})\right|\geq\frac{\epsilon}{2}\right\}\\
&\quad\leq\ 2e^{-2\epsilon^2/4(m(\frac{2}{m})^2+n(\frac{2}{n})^2)}\\
&\quad=\ 2e^{-mn\epsilon^2/8(m+n)}.
\end{aligned}$$

Putting everything together, we get that

$$\begin{aligned}
&\mathbf{P}_{S_X|S_Y=\underline{y}}\left\{\sup_{f\in\mathcal{F}}\left|\hat{A}(f;S)-A(f)\right|\geq\epsilon\right\}\\
&\qquad\leq\ 4\cdot r(\mathcal{F},2m,2n)\cdot e^{-mn\epsilon^2/8(m+n)},
\end{aligned}$$

for $mn\epsilon^2/(m+n)\geq 2$. In the other case, *i.e.*, for $mn\epsilon^2/(m+n)<2$, the bound is greater than one and therefore holds trivially. □

On the Path to an Ideal ROC Curve: Considering Cost Asymmetry in Learning Classifiers

Francis R. Bach*
Computer Science Division
University of California
Berkeley, CA 94720
fbach@cs.berkeley.edu

David Heckerman & Eric Horvitz
Microsoft Research
Redmond, WA 98052
{*heckerman,horvitz*}*@microsoft.com*

Abstract

Receiver Operating Characteristic (ROC) curves are a standard way to display the performance of a set of binary classifiers for all feasible ratios of the costs associated with false positives and false negatives. For linear classifiers, the set of classifiers is typically obtained by training once, holding constant the estimated slope and then varying the intercept to obtain a parameterized set of classifiers whose performances can be plotted in the ROC plane. In this paper, we consider the alternative of varying the asymmetry of the cost function used for training. We show that the ROC curve obtained by varying the intercept and the asymmetry—and hence the slope—always outperforms the ROC curve obtained by varying only the intercept. In addition, we present a path-following algorithm for the support vector machine (SVM) that can compute efficiently the entire ROC curve, that has the same computational properties as training a single classifier. Finally, we provide a theoretical analysis of the relationship between the asymmetric cost model assumed when training a classifier and the cost model assumed in applying the classifier. In particular, we show that the mismatch between the step function used for testing and its convex upper bounds usually used for training leads to a provable and quantifiable difference around extreme asymmetries.

1 INTRODUCTION

Receiver Operating Characteristic (ROC) analysis has seen increasing attention in the recent statistics and machine-learning literature (Pepe, 2000, Provost and Fawcett, 2001, Flach, 2003). The ROC is a representation of choice for displaying the performance of a classifier when the costs assigned by end users to false positives and false negatives are not known at the time of training. For example, when training a classifer for identifying cases of undesirable unsolicited email, end users may have different preferences about the likelihood of a false negative and false positive. The ROC curve for such a classifier reveals the ratio of false negatives and positives at different probability thresholds for classifying an email message as unsolicited or normal email.

In this paper, we consider linear binary classification of points in an Euclidean space—noting that it can be extended in a straightforward manner to non-linear classification problems by using Mercer kernels (Schölkopf and Smola, 2002). That is, given data $x \in \mathbb{R}^d$, $d \geqslant 1$, we consider classifiers of the form $f(x) = \text{sign}(w^\top x + b)$, where $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are referred to as the *slope* and the *intercept*. To date, ROC curves have been usually constructed by training once, holding constant the estimated slope and varying the intercept to obtain the curve. In this paper, we show that, while the latter procedure appears to be the most practical thing to do, it may lead to classifiers with poor performance in some parts of the ROC curve.

The crux of our approach is that we allow the asymmetry of the cost function to vary—i.e., we vary the ratio of the cost of a false positive and the cost of a false negative. For each value of the ratio, we obtain a different slope and intercept, each optimized for this ratio. In a naive implementation, varying the asymmetry would require a retraining of the classifier for each point of the ROC curve, which would be computationally expensive. In Section 3.1, we present an algorithm that can compute the solution of an SVM (Schölkopf and Smola, 2002) for all possible costs of false positives and false negatives, with the same computational complexity as obtaining the solution for only one cost

*This work was done during a summer internship at Microsoft Research.

function. The algorithm extends to asymmetric costs the algorithm of Hastie et al. (2005) and is based on path-following techniques that take advantage of the piecewise linearity of the path of optimal solutions. In Section 3.2, we show how the path-following algorithm can be used to obtain the best possible ROC curve (in expectation). In particular, by allowing both the asymmetry and the intercept to vary, we can obtain provably better ROC curves than by methods that simply vary the intercept.

In Section 4, we provide a theoretical analysis of the link between the asymmetry of costs assumed in training a classifier and the asymmetry desired in its application. In particular, we show that—even in the population (*i.e.*, infinite sample) case—the use of a training loss function which is a convex upper bound on the true or testing loss function (a step function) creates classifiers with sub-optimal accuracy. We quantify this problem around extreme asymmetries for several classical convex-upper-bound loss functions—the square loss and the *erf loss*, an approximation of the logistic loss based on normal cumulative distribution functions (also referred to as the "error function", and usually abbreviated as erf). The analysis is carried through for Gaussian and mixture of Gaussian class-conditional distributions (see Section 4 for more details). As we shall see, the consequences of the potential mismatch between the cost functions assumed in testing versus training underscore the value of using the algorithm that we introduce in Section 4.3. Even when costs are known (i.e., when only one point on the ROC curve is needed), the classifier resulting from our approach which builds the entire ROC curve is never less accurate and can be more accurate than one trained with the known costs using a convex-upper-bound loss function.

2 PROBLEM OVERVIEW

Given data $x \in \mathbb{R}^d$ and labels $y \in \{-1, 1\}$, we consider linear classifiers of the form $f(x) = \text{sign}(w^\top x + b)$, where w is the *slope* of the classifier and b the *intercept*. A classifier is determined by the parameters $(w, b) \in \mathbb{R}^{d+1}$. In Section 2.1, we introduce notation and definitions; in Section 2.2, we lay out the necessary concepts of ROC analysis. In Section 2.3, we describe how these classifiers and ROC curves are typically obtained from data.

2.1 ASYMMETRIC COST AND LOSS FUNCTIONS

Positive (resp. negative) examples are those for which $y = 1$ (resp. $y = -1$). The two types of misclassification, false positives and false negatives, are assigned two different costs, and the total *expected* cost is equal to

$$\begin{aligned} R(C_+, C_-, w, b) &= C_+ P\{w^\top x + b < 0,\ y = 1\} \\ &\quad + C_- P\{w^\top x + b \geqslant 0,\ y = -1\} \end{aligned}$$

If we let $\phi_{0-1}(u) = 1_{u<0}$ be the *0-1 loss*, we can write the expected cost as

$$\begin{aligned} R(C_+, C_-, w, b) &= C_+ E\{1_{y=1}\phi_{0-1}(w^\top x + b)\} \\ &\quad + C_- E\{1_{y=-1}\phi_{0-1}(-w^\top x - b)\} \end{aligned}$$

where E denotes the expectation with respect to the joint distribution of (x, y). The expected cost defined using the 0-1 loss is the cost that end users are usually interested in during the use of the classifier, while the other cost functions that we define below are used solely for training purposes. The convexity of these cost functions makes learning algorithms convergent without local minima, and leads to attractive asymptotic properties (Bartlett et al., 2004).

A traditional set-up for learning linear classifiers from labeled data is to consider a convex upper bound ϕ on the 0-1 loss ϕ_{0-1}, and use the expected ϕ-cost :

$$\begin{aligned} R_\phi(C_+, C_-, w, b) &= C_+ E\{1_{y=1}\phi(w^\top x + b)\} \\ &\quad + C_- E\{1_{y=-1}\phi(-w^\top x - b)\} \end{aligned}$$

We refer to the ratio $C_+/(C_- + C_+)$ as the *asymmetry*. We shall use *training asymmetry* to refer to the asymmetry used for training a classifier using a ϕ-cost, and the *testing asymmetry* to refer to the asymmetric cost characterizing the testing situation (reflecting end user preferences) with the actual cost based on the 0-1 loss. In Section 4, we will show that these may be different in the general case.

We shall consider several common loss functions. Some of the loss functions (square loss, hinge loss) lead to attractive computational properties, while others (square loss, erf loss) are more amenable to theoretical manipulations (see Figure 1 for the plot of the loss functions, as they are commonly used and defined below[1]):

- **square loss** : $\phi_{sq}(u) = \frac{1}{2}(u-1)^2$; the classifier is equivalent to linear regression,
- **hinge loss** : $\phi_{hi}(u) = \max\{1-u, 0\}$; the classifier is the support vector machine (Schölkopf and Smola, 2002),
- **erf loss** : $\phi_{erf}(u) = 2\left[\frac{u}{2}\psi\left(\frac{u}{2}\right) - \frac{u}{2} + \psi'\left(\frac{u}{2}\right)\right]$, where ψ is the cumulative distribution of the standard normal distribution, i.e. : $\psi(v) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{v} e^{-t^2/2}dt$, and $\psi'(v) = \frac{1}{\sqrt{2\pi}}e^{-v^2/2}$.

[1]Note that by rescaling, all of these loss functions can be made to be an upper bound on the 0-1 loss which is tight at zero.

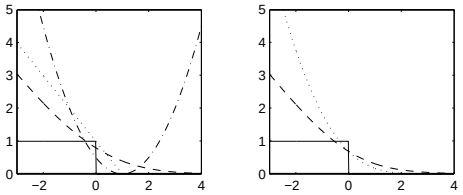


Figure 1: Loss functions: (left) plain: 0-1 loss, dotted: hinge loss, dashed: erf loss, dash-dotted: square loss. (Right) plain: 0-1 loss, dotted: probit loss, dashed: logistic loss.

The erf loss provides a good approximation of the *logistic loss* $\log(1+e^{-u})$ as well as its derivative, and is amenable to closed-form computations for Gaussians and mixture of Gaussians (see Section 4 for more details). Note that the erf loss is different from the *probit loss* $-\log\psi(u)$, which leads to probit regression (Hastie et al., 2001).

2.2 ROC ANALYSIS

The aim of ROC analysis is to display in a single graph the performance of classifiers for all possible costs of misclassification. In this paper, we consider sets of classifiers $f_\gamma(x)$, parameterized by a variable $\gamma \in \mathbb{R}$ (γ can either be the intercept or the training asymmetry).

For a classifier $f(x)$, we can define a point (u, v) in the "ROC plane," where u is the proportion of false positives $u = P(f(x) = 1|y = -1)$, and v is the proportion of true positives $v = P(f(x) = 1|y = 1)$.

When γ is varied, we obtain a curve in the ROC plane, the ROC curve (see Figure 2 for an example). Whether γ is the intercept or the training asymmetry, the ROC curve always passes through the point $(0, 0)$ and $(1, 1)$, which corresponds to classifying all points as negative (resp. positive).

The upper convex envelope of the curve is the set of optimal ROC points that can be achieved by the set of classifiers; indeed, if a point in the envelope is not one of the original points, it must lie in a segment between two points $(u(\gamma_0), v(\gamma_0))$ and $(u(\gamma_1), v(\gamma_1))$, and all points in a segment between two classifiers can always be attained by choosing randomly between the two classifiers (note that this classifier itself is not a linear classifier; this performance can only be achieved by a mixture of two linear classifiers).

Denoting $p_+ = P(y = 1)$ and $p_- = P(y = -1)$, the expected (C_+, C_-)-cost for a classifier (u, v) in the ROC space, is simply $p_+C_+(1 - v) + p_-C_-u$, and thus optimal classifiers for the (C_+, C_-)-cost can be found by looking at lines of slope that are normal to $(p_-C_-, -p_+C_+)$—and thus proportional to

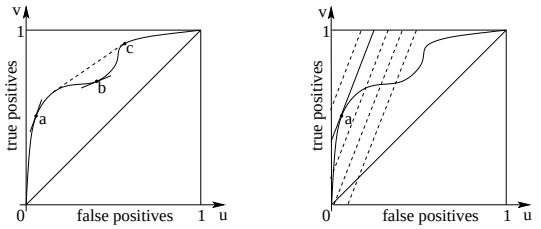


Figure 2: (Left) ROC curve: (plain) regular ROC curve; (dashed) convex envelope. The points a and c are ROC-consistent and the point b is ROC-inconsistent. (Right) ROC curve and dashed equi-cost lines: All lines have direction (p_+C+, p_-C_-), the plain line is optimal and the point "a" is the optimal classifier.

(p_+C_+, p_-C_-)—and which intersects the ROC curve and are as close as the point $(0, 1)$ as possible (see Figure 2).

A point $(u(\gamma), v(\gamma))$ is said to be *ROC-consistent* if it lies on the upper convex envelope; In this case, the tangent direction $(du/d\gamma, dv/d\gamma)$ defines a cost $(C_+(\gamma), C_-(\gamma))$ for which the classifier is optimal (for the testing cost, which is defined using the 0-1 loss), by having $(p_+C_+(\gamma), p_-C_-(\gamma))$ proportional to $(du/d\gamma, dv/d\gamma)$. This leads to an *optimal testing asymmetry* $\beta(\gamma)$, defined as $\beta(\gamma) = \frac{C_+(\gamma)}{C_+(\gamma)+C_-(\gamma)} = \frac{1}{1+\frac{p_+}{p_-}\frac{dv}{d\gamma}(\gamma)/\frac{du}{d\gamma}(\gamma)}$.

If a point $(u(\gamma), v(\gamma))$ is ROC-inconsistent, then the quantity $\beta(\gamma)$ has no meaning, and such a classifier is generally useless, because, for all settings of the misclassification cost, that classifier can be outperformed by the other classifiers or a combination of classifiers.

In Section 4, we relate the optimal asymmetry of cost in the testing or eventual use of a classifer in the real world, to the asymmetry of cost used to train that classifier; in particular, we show that they differ and quantify this difference for extreme asymmetries (*i.e.*, close to the corner points $(0, 0)$ and $(1, 1)$). This analysis highlights the value of generating the entire ROC curve, even when only one point is needed, as we will present in Section 4.3.

2.3 LEARNING FROM DATA

Given n labelled data points (x_i, y_i), the *empirical cost* is equal to:

$$\widehat{R}(C_+, C_-, w, b) = \frac{C_+}{n}\#\{y_i(w^\top x_i + b) < 0, y_i = 1\} + \frac{C_-}{n}\#\{y_i(w^\top x_i + b) < 0, y_i = -1\}$$

while the *empirical ϕ-cost* is equal to

$$\widehat{R}_\phi(C_+, C_-, w, b) = \frac{C_+}{n}\sum_{i\in\mathcal{I}_+}\phi(y_i(w^\top x_i + b)) + \frac{C_-}{n}\sum_{i\in\mathcal{I}_-}\phi(y_i(w^\top x_i + b)),$$

where $\mathcal{I}_+ = \{i, y_i = 1\}$ and $\mathcal{I}_- = \{i, y_i = -1\}$. When learning a classifier from data, a classical setup is to minimize the sum of the *empirical ϕ-cost* and a regularization term $\frac{1}{2n}||w||^2$, *i.e.*, to minimize $\widehat{J}_\phi(C_+, C_-, w, b) = \widehat{R}_\phi(C_+, C_-, w, b) + \frac{1}{2n}||w^2||$.

Note that the objective function is no longer homogeneous in (C_+, C_-); the sum $C_+ + C_-$ is referred to as the total amount of regularization. Thus, two end-user–defined parameters are needed to train a linear classifier: the *total amount of regularization* $C_+ + C_- \in \mathbb{R}^+$, and the *asymmetry* $\frac{C_+}{C_+ + C_-} \in [0, 1]$. In Section 3.1, we show how the minimum of $\widehat{J}_\phi(C_+, C_-, w, b)$, with respect to w and b, can be computed efficiently for the hinge loss, for many values of (C_+, C_-), with a computational cost that is within a constant factor of the computational cost of obtaining a solution for one value of (C_+, C_-).

Building an ROC curve from data If a sufficiently large validation set is available, we can train on the training set and use the empirical distribution of the validation data to plot the ROC curve. If sufficient validation data is not available, then we can use 10 random half splits of the data, train a classifier on one half and use the other half to obtain the ROC scores. Then, for each value of the parameter γ that defines the ROC curve (either the intercept or the training asymmetry), we average the 10 scores. We can also use this approach to obtain confidence intervals (Flach, 2003).

3 BUILDING ROC CURVES FOR THE SVM

In this section, we will present an algorithm to compute ROC curves for the SVM that explores the two-dimensional space of cost parameters (C_+, C_-) efficiently. We first show how to obtain optimal solutions of the SVM without solving the optimization problems many times for each value of (C_+, C_-). This method generalizes the results of Hastie et al. (2005) to the case of asymmetric cost functions. We then describe how the space (C_+, C_-) can be appropriately explored and how ROC curves can be constructed.

3.1 BUILDING PATHS OF CLASSIFIERS

Given n data points x_i, $i = 1, \ldots, n$ which belong to $\mathbb{R}^d$, and n labels $y_i \in \{-1, 1\}$, minimizing the regularized empirical hinge loss is equivalent to solving the following convex optimization problem (Schölkopf and Smola, 2002):

$$\min_{w,b,\xi} \sum_i C_i \xi_i + \frac{1}{2}||w||^2 \quad \text{s.t.} \quad \forall i,\ \xi_i \geqslant 0, \quad \forall i,\ \xi_i \geqslant 1 - y_i(w^\top x_i + b)$$

where $C_i = C_+$ if $y_i = 1$ and $C_i = C_-$ if $y_i = -1$.

The dual problem is the following:

$$\max_{\alpha \in \mathbb{R}^n} -\frac{1}{2}\alpha^\top \operatorname{Diag}(y) K \operatorname{Diag}(y) \alpha + 1^\top \alpha \quad \text{s.t. } \alpha^\top y = 0 \text{ and } \forall i,\ 0 \leqslant \alpha_i \leqslant C_i$$

Following Hastie et al. (2005), from the KKT optimality conditions, for an optimal set of primal-dual variables (w, b, α), we can separate data points into three disjoint sets: $\mathcal{M} = \{i, \alpha_i \in [0, C_i], y_i(w^\top x_i + b) = 1\}$, $\mathcal{L} = \{i, \alpha_i = C_i, y_i(w^\top x_i + b) < 1\}$, $\mathcal{R} = \{i, \alpha_i = 0, y_i(w^\top x_i + b) > 1\}$.

These sets are usually referred to as *active sets* (Boyd and Vandenberghe, 2003). If the sets $\mathcal{M}$, $\mathcal{L}$ and $\mathcal{R}$ are known, then it is straightforward to show that, in the domain where the active sets remain constant, the optimal primal-dual variables (w, α, b) are affine functions of (C_+, C_-). This implies that the optimal variables (w, α, b) are piecewise affine continuous functions of the vector (C_+, C_-), with "kinks" where the active sets change.

Following a path The active sets remain the same as long as all constraints defining the active sets are satisfied, *i.e.*, (a) $y_i(w^\top x_i + b) - 1$ is positive for all $i \in \mathcal{R}$ and negative for all $i \in \mathcal{L}$, and (b) for each $i \in \mathcal{M}$, α_i remains between 0 and C_i. This defines a set of linear inequalities in (C_+, C_-). The facets of the polytope defined by these inequalities can always be found in linear time in n, when efficient convex hull algorithms are used (Avis et al., 1997). However, when we only follow a straight line in the (C_+, C_-)-space, the polytope is then a segment and its extremities are trivial to find (also in $O(n)$).

Following Hastie et al. (2005), if a solution is known for one value of (C_+, C_-), we can follow the path along a line, by simply monitoring which constraint is violated first and changing the active sets accordingly.

Path initialization Hastie et al. (2005) requires that the training datasets are balanced in order to avoid solving a quadratic program to enter the path, *i.e.*, if n_+ (resp. n_-) is the number of positive (resp. negative) training examples, then they require that $n_+ = n_-$. In our situation, we can explore the two dimensional space (C_+, C_-). Thus, the requirements become $C_+ n_+ = C_- n_-$ along the path. We can start the path by following the line $C_+ n_+ = C_- n_-$ and avoid the need to solve a quadratic program.

Computational complexity As shown by Hastie et al. (2005), if the appropriate online linear algebra tools are used, the complexity of obtaining one path of classifiers across one line is the same as obtaining the solution for one SVM using classical techniques such as sequential minimal optimization (Platt, 1998).

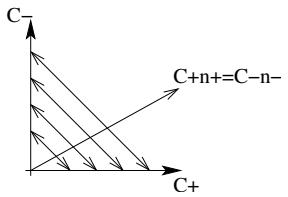


Figure 3: Lines in the (C_+, C_-)-space. The line $C_+n_+ = C_-n_-$ is always followed first; then several lines with constant $C_+ + C_-$ are followed in parallel, around the optimal line for the validation data (bold curve).

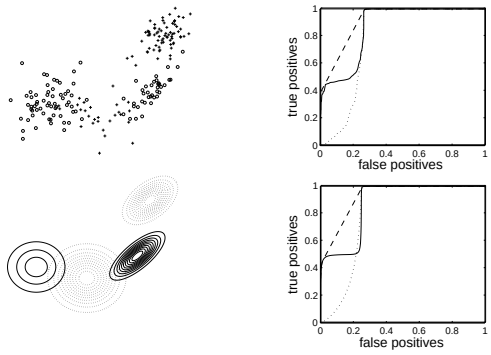


Figure 4: Two examples of ROC curves for bimodal class conditional densities, varying intercept (dotted), varying asymmetry (plain) and varying both (dashed). (Top) obtained from 10 random splits, using the data shown on the left side (one class is plotted as circles, the other one as crosses), (Bottom) obtained from population densities (one class with plain density contours, the other one with dotted contours).

3.2 CONSTRUCTING THE ROC CURVE

Given the tools of Section 3.1, we can learn paths of linear classifiers from data. In this section, we present an algorithm to build ROC curves from the paths. We do this by exploring relevant parts of the (C_+, C_-) space, selecting the best classifiers among the ones that are visited.

We assume that we have two separate datasets, one for training and one for testing. This approach generalizes to cross-validation settings in a straightforward manner.

Exploration phase In order to start the path-following algorithm, we need to start at $C_+ = C_- = 0$ and follow the line $C_+n_+ = C_-n_-$. We follow this line up to a large upper bound on $C_+ + C_-$. For all classifiers along that line, we compute a misclassification cost on the testing set, with given asymmetry (C_+^0, C_-^0) (as given by the user, and usually, but not necessarily, close to a point of interest in the ROC space). We then compute the best performing pair (C_+^1, C_-^1) and we select pairs of the form (rC_+^1, rC_-^1), where r belongs to a set R of the type $R = \{1, 10, 1/10, 100, 1/100, \dots\}$. The set R provides further explorations for the total amount of regularization $C_+ + C_-$.

Then, for each r, we follow the paths of direction $(1, -1)$ and $(-1, 1)$ starting from the point (rC_+^1, rC_-^1). Those paths have a fixed total amount of regularization but vary in asymmetry. In Figure 3, we show all of lines that are followed in the (C_+, C_-) space.

Selection phase After the exploration phase, we have $|R| + 1$ different lines in the (C_+, C_-) space: the line $C_-n_- = C_+n_+$, and the $|R|$ lines $C_+ + C_- = r(C_+^1 + C_-^1), r \in R$. For each of these lines, we know the optimal solution (w, b) for any cost settings on that line. The line $C_-n_- = C_+n_+$ is used for computational purposes (*i.e.*, to enter the path), so we do not use it for testing purposes.

From R lines in the (C_+, C_-)-plane, we build the three ROC curves shown in Figure 4, for a finite sample problem and for an infinite sample problem (for the infinite sample, the solution of the SVM was obtained by working directly with densities):

- *Varying intercept*: we extract the slope w corresponding to the best setting $(C_+^1 + C_-^1)$, and vary the intercept b from $-\infty$ to ∞. This is the traditional method for building an ROC curve for an SVM.
- *Varying asymmetry*: we only consider the line $C_+ + C_- = C_+^1 + C_-^1$ in the (C_+, C_-)-plane; the classifiers that are used are the optimal solutions of the convex optimization problem. Note that for each value of the asymmetry, we obtain a different value of the slope and the intercept.
- *Varying intercept and asymmetry*: for each of the points on the R lines in the (C_+, C_-)-plane, we discard the intercept b and keep the slope w obtained from the optimization problem; we then use all possible intercept values; this leads to R two-dimensional surfaces in the ROC plane. We then compute the convex envelope of these, to obtain a single curve.

Since all classifiers obtained by varying only the intercept (resp. the asymmetry) are included in the set used for varying both the intercept and the asymmetry, the third ROC curve always outperforms the first two curves (*i.e.*, it is always closer to the top left corner). This is illustrated in Figure 4.

Intuitively, the ROC curve obtained by varying the asymmetry should be better than the ROC generated by varying the intercept because, for each point, the slope of the classifier is optimized. Empirically, this is generally true, but is not always the case, as displayed

in Figure 4. This is not a small sample effect, as the infinite sample simulation shows. Another troubling fact is that the ROC curve obtained by varying asymmetry, exhibits strong concavities, *i.e.*, there are many ROC-inconsistent points: for those points, the solution of the SVM with the corresponding asymmetry is far from being the best linear classifier when performance is measured with the same asymmetry but with the exact 0-1 loss. In addition, even for ROC-consistent points, the training asymmetry and the testing asymmetry differ. In the next section, we analyze why they may differ and characterize their relationships in some situations.

4 TRAINING VS. TESTING ASYMMETRY

We observed in Section 3.2 that the training cost asymmetry can differ from the testing asymmetry. In this section, we analyze their relationships more closely for the population (*i.e.*, infinite sample) case. Although a small sample effect might alter some of the results presented in this section, we argue that most of the discrepancies come from using a convex surrogate to the $0-1$ loss.

The *Bayes optimal* classifier for a given asymmetry (C_+, C_-), is the (usually non-linear) classifier with minimal expected cost. A direct consequence of results in Bartlett et al. (2004) is that, if the Bayes optimal classifier is linear, then using a convex surrogate has no effect, *i.e.*, using the expected ϕ-cost will lead to the minimum expected cost. Thus, if the Bayes optimal classifier is linear, then, in the population case (infinite sample), there should be no difference. However, when the Bayes optimal classifier is not linear, then we might expect to obtain a difference, and we demonstrate that we do have one and quantify it for several situations.

Since we are using population densities, we can get rid of the regularization term and thus only the asymmetry will have an influence on the results, *i.e.*, we can restrict ourselves to $C_+ + C_- = 1$. We let $\gamma = C_+/(C_+ + C_-) = C_+$ denote the training asymmetry. For a given training asymmetry γ and a loss function ϕ, in Section 2.2, we defined the *optimal testing asymmetry* $\beta(\gamma)$ for the training asymmetry γ. In this section, we will refer to the $\beta(\gamma)$ simply as the *testing asymmetry*.

Although a difference might be seen empirically for all possible asymmetries, we analyze the relationship between the testing cost asymmetry and training asymmetry in cases of extreme asymmetry, *i.e.*, in the ROC framework, close to the corner points $(0,0)$ and $(1,1)$. We prove that, depending on the class conditional densities, there are two possible different regimes for extreme asymmetries: either the optimal testing asymmetry is more extreme, or it is less extreme. We also provide, under certain conditions, a simple test that can determine the regime given class conditional densities.

In this section, we choose class conditional densities that are either Gaussian or a mixture of Gaussians, because (a) any density can be approximated as well as desired by mixtures of Gaussians (Hastie et al., 2001), and (b) for the square loss and the erf loss, they enable closed-form calculations that lead to Taylor expansions.

4.1 OPTIMAL SOLUTIONS FOR EXTREME COST ASYMMETRIES

We assume that the class conditional densities are mixtures of Gaussian, *i.e.*, the density of positive (resp. negative) examples is a mixture of k_+ Gaussians, with means μ_+^i and covariance matrix Σ_+^i, and mixing weights π_i^+, $i \in \{1, \ldots, m_+\}$ (resp. k_- Gaussians, with means μ_-^i and covariance matrix Σ_-^i, and mixing weights π_-^+, $i \in \{1, \ldots, m_-\}$). We denote p_+ and p_- as the marginal class densities, $p_+ = P(y=1)$, $p_- = P(y=-1)$. We assume that all mixing weights $\pi_\pm^i$ are strictly positives and that all covariance matrices $\Sigma_\pm^i$ have full rank.

In the following sections, we provide Taylor expansions of various quantities around the null training asymmetry $\gamma = 0$. They trivially extend around the reverse asymmetry $\gamma = 1$. We start with an expansion of the unique global minimum (w, b) of the ϕ-cost with asymmetry γ. For the square loss, (w, b) can be obtained in closed form for any class conditional densities so the expansion is easy to obtain, while for the erf loss, an asymptotic analysis of the optimality conditions has to be carried through, and is only valid for mixture of Gaussians (see Bach et al. (2004) for proofs).

Proposition 1 (square loss) *Under previous assumptions, we have the following expansions:*

$$\begin{aligned} w &= 2\frac{p_+}{p_-}\gamma\Sigma_-^{-1}(\mu_+ - \mu_-) + O(\gamma^2) \\ b &= -1 + \frac{p_+}{p_-}\gamma[2 - 2\mu_-^\top(\mu_+ - \mu_-)] + O(\gamma^2) \end{aligned}$$

where $m = \mu_+ - \mu_-$, and $\Sigma_\pm$ and $\mu_\pm$ are the class conditional means and variances.

Proposition 2 (erf loss) *Under previous assumptions, we have the following expansions:*

$$\begin{aligned} w &= (2\log(1/\gamma))^{-1/2}\,\widetilde{\Sigma}_-(\tilde{\mu}_+ - \tilde{\mu}_-) + o\left(\log(1/\gamma)^{-1/2}\right) \\ b &= -(2\log(1/\gamma))^{1/2} + o\left(\log(1/\gamma)^{1/2}\right) \end{aligned}$$

where $\tilde{m} = \tilde{\mu}_+ - \tilde{\mu}_-$, *and* $\widetilde{\Sigma}_\pm$ *and* $\tilde{\mu}_\pm$ *are convex combinations of the mixture means and covariances,* i.e., *there exists strictly positive weights* $\tilde{\pi}_\pm^i$, *that sum to one for each sign, such that* $\widetilde{\Sigma}_\pm = \sum_i \tilde{\pi}_\pm^i \Sigma_\pm^i$ *and* $\tilde{\mu}_\pm = \sum_i \tilde{\pi}_\pm^i \mu_\pm^i$.

The weights $\tilde{\pi}_\pm^i$ can be expressed as the solution of a convex optimization problem (see Bach et al. (2004) for more details). When there is only one mixture component (Gaussian densities), then $\tilde{\pi}_\pm^1 = 1$.

4.2 EXPANSION OF TESTING ASYMMETRIES

Using the expansions of Proposition 1 and 2, we can readily derive an expansion of the ROC coordinates for small γ, as well as the testing asymmetry $\beta(\gamma)$. We have (see Bach et al. (2004) for proofs):

Proposition 3 (square loss) *Under previous assumptions, we have the following expansion:*

$$\log\left(\frac{p_-}{p+}(\beta(\gamma)^{-1}-1)\right) = \frac{p_-^2}{8p_+^2\gamma^2}\left(\frac{1}{m^\top \Sigma_-^{-1}\Sigma_-^{i_-}\Sigma_-^{-1}m} - \frac{1}{m^\top \Sigma_-^{-1}\Sigma_+^{i_+}\Sigma_-^{-1}m}\right) + o(1/\gamma^2) \quad (1)$$

where i_- *(resp.* i_+*) is one of the negative (resp. positive) mixture component.*

Proposition 4 (erf loss) *Under previous assumptions, we have the following expansion:*

$$\log\left(\frac{p_-}{p+}(\beta(\gamma)^{-1}-1)\right) = 2\log(1/\gamma)\left(\frac{1}{\tilde{m}^\top \widetilde{\Sigma}_-^{-1}\Sigma_-^{i_-}\widetilde{\Sigma}_-^{-1}\tilde{m}} - \frac{1}{\tilde{m}^\top \widetilde{\Sigma}_-^{-1}\Sigma_+^{i_+}\widetilde{\Sigma}_-^{-1}\tilde{m}}\right) + o(\log(1/\gamma)) \quad (2)$$

where i_- *(resp.* i_+*) is one of the negative (resp. positive) mixture component.*

The rest of the analysis is identical for both losses and thus, for simplicity, we focus primarily on the square loss. For the square loss, we have two different regimes, depending on the sign of $m^\top \Sigma_-^{-1}\Sigma_-^{i_-}\Sigma_-^{-1}m - m^\top \Sigma_-^{-1}\Sigma_+^{i_+}\Sigma_-^{-1}m$:

- if $m^\top \Sigma_-^{-1}\Sigma_-^{i_-}\Sigma_-^{-1}m > m^\top \Sigma_-^{-1}\Sigma_+^{i_+}\Sigma_-^{-1}m$, then from the expansion in Eq. (1) and Eq. (2), we see that the testing asymmetry tends to 1 exponentially fast. Because this is an expansion around the null training asymmetry, the ROC curve must be starting on the bottom right part of the main diagonal and the points close to $\gamma = 0$ are not ROC-consistent, *i.e.*, the classifiers with training asymmetry too close to zero are useless as they are too extreme.

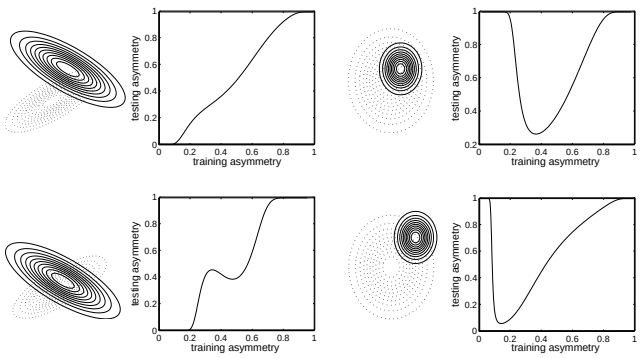


Figure 5: Training asymmetry vs. testing asymmetry, **square loss**: (Left) Gaussian class conditional densities, (right) testing asymmetry vs. training asymmetry; from top to bottom, the values of $(m^\top \Sigma_-^{-1}\Sigma_-^{i_-}\Sigma_-^{-1}m)^{-1} - (m^\top \Sigma_-^{-1}\Sigma_+^{i_+}\Sigma_-^{-1}m)^{-1}$ are 0.12, -6, 3, -0.96.

- if $m^\top \Sigma_-^{-1}\Sigma_-^{i_-}\Sigma_-^{-1}m < m^\top \Sigma_-^{-1}\Sigma_+^{i_+}\Sigma_-^{-1}m$, then from the expansion in Eq. (1) and Eq. (2), we see that the testing asymmetry tends to 0 exponentially fast, in particular, the derivative $d\beta/d\gamma$ is null at $\gamma = 0$, meaning, that the testing asymmetry is significantly smaller than the training asymmetry, *i.e.*, less extreme.
- if $m^\top \Sigma_-^{-1}\Sigma_-^{i_-}\Sigma_-^{-1}m = m^\top \Sigma_-^{-1}\Sigma_+^{i_+}\Sigma_-^{-1}m$, then the asymptotic expansion does not provide any information relating to the behavior of the testing asymmetry. We are currently investigating higher-order expansions in order to study the behavior of this limiting case. Note that when the two class conditional densities are Gaussians with identical covariance (a case where the Bayes optimal classifier with symmetric cost is indeed linear), we are in the present case.

The strength of the effects we have described above depends on the norm of $m = \mu_+ - \mu_-$: if m is large, *i.e.*, the classification problem is simple, then those effects are less strong, while when m is small, they are stronger. In Figure 5, we provide several examples for the square loss, with the two regimes and different strengths. It is worth noting, that, although the theoretical results obtained in this section are asymptotic expansions around the corners (*i.e.*, extreme asymmetries), the effects also remain valid far from the corners.

We thus must test to identify which regime we are in, namely testing for the sign of $m^\top \Sigma_-^{-1}\Sigma_-^{i_-}\Sigma_-^{-1}m = m^\top \Sigma_-^{-1}\Sigma_+^{i_+}\Sigma_-^{-1}m$. This test requires knowledge of the class conditional densities; it can currently always be performed in closed form for the square loss, while for the erf loss, it requires to solve a convex optimization problem, described by Bach et al. (2004).

Dataset	γ	one asym.	all asym.
PIMA	0.68	41 ± 0.4	22 ± 1
BREAST	0.99	0.9 ± 0.03	0.09 ± 0.04
IONOSPHERE	0.82	10 ± 0.5	4 ± 0.8
LIVER	0.32	27 ± 1.8	23.8 ± 0.02
RINGNORM	0.94	6.3 ± 0.06	4.3 ± 0.1
TWONORM	0.16	15 ± 0.2	1.2 ± 0.2
ADULT	0.70	12.8 ± 0.8	11.5 ± 0.3

Table 1: Training with the testing asymmetry γ vs. training with all cost asymmetries: we report validation costs obtained from 10 half-random splits (premultiplied by 100). Only the asymmetry with the largest difference is reported. Given an asymmetry γ we use the cost settings $C_+ = 2\gamma$, $C_- = 2(1-\gamma)$ (which leads to the misclassification error if $\gamma = 1/2$).

4.3 BUILDING THE ENTIRE ROC CURVE FOR A SINGLE POINT

As shown empirically in Section 3.2, and demonstrated theoretically in this section, training and testing asymmetries differ; and this difference suggests that even when the user is interested in only one cost asymmetry, the training procedure should explore more cost asymmetries, i.e. build the ROC curve as presented in Section 3.2 and chose the best classifier as follows: a given asymmetry in cost for the test case leads to a unique slope in the ROC space, and the optimal point for this asymmetry is the point on the ROC curve whose tangent has the corresponding slope and which is closest to the upper-left corner.

We compare in Figure 1, for various datasets and linear classifiers, the performance with cost asymmetry γ of training a classifier with cost asymmetry γ to the performance of training with all cost asymmetries. Using all asymmetries always perform better than assuming a single asymmetry—we simply have more classifiers to choose from. In Figure 1, we report only the performance for the cost asymmetries which show the greatest differences, showing that in some cases, it is very significant, and that a simple change in the training procedure may lead to substantial gains.

5 CONCLUSION

We have presented an efficient algorithm to build ROC curves by varying the training cost asymmetries for SVMs. The algorithm is based on the piecewise linearity of the path of solutions when the cost of false positives and false negatives vary. We have also provided a theoretical analysis of the relationship between the potentially different cost asymmetries assumed in training and testing classifiers, showing that they differ under certain circumstances. We have characterized key relationships, and have worked to highlight the potential value of building the entire ROC curve even when a single point may be needed. Such an approach can lead to a significant improvement of performance with little added computational cost. Finally, we note that, although we have focused in this paper on the single kernel learning problem, our approach can be readily extended to the multiple kernel learning setting (Bach et al., 2005) with appropriate numerical path following techniques.

Acknowledgments

We thank John Platt for sharing his experiences and insights with considerations of cost functions in training and testing support vector machines.

References

D. Avis, D. Bremner, and R. Seidel. How good are convex hull algorithms ? In *Computational Geometry: Theory and Applications*, volume 7, 1997.

F. R. Bach, D. Heckerman, and E. Horvitz. On the path to an ideal ROC curve: Considering cost asymmetry in learning classifiers. Technical Report MSR-TR-2004-124, Microsoft Research, 2004.

F. R. Bach, R. Thibaux, and M. I. Jordan. Computing regularization paths for learning multiple kernels. In *NIPS 17*, 2005.

P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Large margin classifiers: convex loss, low noise, and convergence rates. In *NIPS 16*, 2004.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge Univ. Press, 2003.

P. A. Flach. The geometry of ROC space: understanding machine learning metrics through ROC isometrics. In *ICML*, 2003.

T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *JMLR*, 5:1391–1415, 2005.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

M. S. Pepe. Receiver operating characteristic methodology. *J. Am. Stat. Assoc.*, 95(449):308–311, 2000.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods: Support Vector Learning*, Cambridge, MA, 1998. MIT Press.

F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3): 203–231, 2001.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

On Manifold Regularization

Mikhail Belkin, Partha Niyogi, Vikas Sindhwani
{misha,niyogi,vikass}@cs.uchicago.edu
Department of Computer Science
University of Chicago

Abstract

We propose a family of learning algorithms based on a new form of regularization that allows us to exploit the geometry of the marginal distribution. We focus on a semi-supervised framework that incorporates labeled and unlabeled data in a general-purpose learner. Some transductive graph learning algorithms and standard methods including Support Vector Machines and Regularized Least Squares can be obtained as special cases. We utilize properties of Reproducing Kernel Hilbert spaces to prove new Representer theorems that provide theoretical basis for the algorithms. As a result (in contrast to purely graph based approaches) we obtain a natural out-of-sample extension to novel examples and are thus able to handle both transductive and truly semi-supervised settings. We present experimental evidence suggesting that our semi-supervised algorithms are able to use unlabeled data effectively. In the absence of labeled examples, our framework gives rise to a regularized form of spectral clustering with an out-of-sample extension.

1 Introduction

The problem of learning from labeled and unlabeled data (*semi-supervised* and *transductive* learning) has attracted considerable attention in recent years (cf. [11, 7, 10, 15, 18, 17, 9]). In this paper, we consider this problem within a new framework for data-dependent regularization. Our framework exploits the geometry of the probability distribution that generates the data and incorporates it as an additional regularization term. We consider in some detail the special case where this probability distribution is supported on a submanifold of the ambient space.

Within this general framework, we propose two specific families of algorithms: the Laplacian Regularized Least Squares (hereafter LapRLS) and the Laplacian Support Vector Machines (hereafter LapSVM). These are natural extensions of RLS and SVM respectively. In addition, several recently proposed transductive methods (e.g., [18, 17, 1]) are also seen to be special cases of this general approach. Our solution for the semi-supervised case can be expressed as an expansion over labeled and unlabeled data points. Building on a solid theoretical foundation, we obtain a natural solution to the problem of out-of-sample extension (see also [6] for some recent work).When all examples are unlabeled, we obtain a new regularized version of spectral clustering.

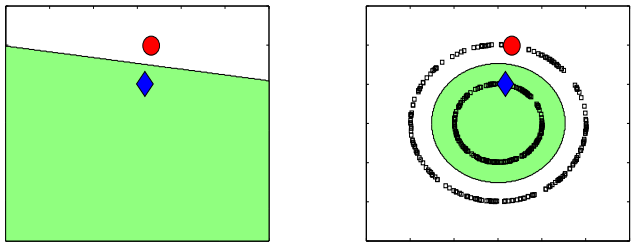
Our general framework brings together three distinct concepts that have received some independent recent attention in machine learning: Regularization in Reproducing Kernel Hilbert Spaces, the technology of Spectral Graph Theory and the geometric viewpoint of Manifold Learning algorithms.

2 The Semi-Supervised Learning Framework

First, we recall the standard statistical framework of learning from examples, where there is a probability distribution P on $X \times \mathbb{R}$ according to which training examples are generated. Labeled examples are (x, y) pairs drawn from P. Unlabeled examples are simply $x \in X$ drawn from the marginal distribution $\mathcal{P}_X$ of P.

One might hope that knowledge of the marginal $\mathcal{P}_X$ can be exploited for better function learning (e.g. in classification or regression tasks). Figure 1 shows how unlabeled data can radically alter our prior belief about the appropriate choice of classification functions. However, if there is no identifiable relation be-

Figure 1: Unlabeled data and prior beliefs



tween $\mathcal{P}_X$ and the conditional $\mathcal{P}(y|x)$, the knowledge of $\mathcal{P}_X$ is unlikely to be of much use. Therefore, we will make a specific assumption about the connection between the marginal and the conditional. We will assume that if two points $x_1, x_2 \in X$ are *close* in the *intrinsic* geometry of $\mathcal{P}_X$, then the conditional distributions $\mathcal{P}(y|x_1)$ and $\mathcal{P}(y|x_2)$ are similar. In other words, the conditional probability distribution $\mathcal{P}(y|x)$ varies smoothly along the geodesics in the intrinsic geometry of $\mathcal{P}_X$.

We utilize these geometric ideas to extend an established framework for function learning. A number of popular algorithms such as SVM, Ridge regression, splines, Radial Basis Functions may be broadly interpreted as regularization algorithms with different empirical cost functions and complexity measures in an appropriately chosen Reproducing Kernel Hilbert Space (RKHS).

For a Mercer kernel $K : X \times X \to \mathbb{R}$, there is an associated RKHS $\mathcal{H}_K$ of functions $X \to \mathbb{R}$ with the corresponding norm $\| \|_K$. Given a set of labeled examples (x_i, y_i), $i = 1, \ldots, l$ the standard framework estimates an unknown function by minimizing

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma \|f\|_K^2 \qquad (1)$$

where V is some loss function, such as squared loss $(y_i - f(x_i))^2$ for RLS or the soft margin loss function for SVM. Penalizing the RKHS norm imposes smoothness conditions on possible solutions. The classical Representer Theorem states that the solution to this minimization problem exists in $\mathcal{H}_K$ and can be written as

$$f^*(x) = \sum_{i=1}^{l} \alpha_i K(x_i, x) \qquad (2)$$

Therefore, the problem is reduced to optimizing over the finite dimensional space of coefficients α_i, which is the algorithmic basis for SVM, Regularized Least Squares and other regression and classification schemes.

Our goal is to extend this framework by incorporating additional information about the geometric structure of the marginal $\mathcal{P}_X$. We would like to ensure that the solution is smooth with respect to both the ambient space and the marginal distribution $\mathcal{P}_X$. To achieve that, we introduce an additional regularizer:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A \|f\|_K^2 + \gamma_I \|f\|_I^2 \qquad (3)$$

where $\|f\|_I^2$ is an appropriate penalty term that should reflect the intrinsic structure of $\mathcal{P}_X$. Here γ_A controls the complexity of the function in the *ambient* space while γ_I controls the complexity of the function in the *intrinsic* geometry of $\mathcal{P}_X$. Given this setup one can prove the following representer theorem:

Theorem 2.1. *Assume that the penalty term $\|f\|_I$ is sufficiently smooth with respect to the RKHS norm $\|f\|_K$. Then the solution f^* to the optimization problem in Eqn 3 above exists and admits the following representation*

$$f^*(x) = \int_{\mathcal{M}} \alpha(y) K(x, y)\, d\mathcal{P}_X(y) + \sum_{i=1}^{l} \alpha_i K(x_i, x) \qquad (4)$$

where $\mathcal{M} = \operatorname{supp}\{\mathcal{P}_X\}$ is the support of the marginal $\mathcal{P}_X$.

The proof of this theorem runs over several pages and is omitted for lack of space. See [4] for details including the exact statement of the smoothness conditions.

In most applications, however, we do not know $\mathcal{P}_X$. Therefore we must attempt to get empirical estimates of $\|f\|_I$. Note that in order to get such empirical estimates it is sufficient to have *unlabeled* examples.

A case of particular recent interest is when the support of $\mathcal{P}_X$ is a compact submanifold $\mathcal{M} \subset X = \mathbb{R}^n$. In that case, a natural choice for $\|f\|_I$ is $\int_{\mathcal{M}} \langle \nabla_{\mathcal{M}} f, \nabla_{\mathcal{M}} f \rangle$. The optimization problem becomes

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A \|f\|_K^2 + \gamma_I \int_{\mathcal{M}} \langle \nabla_{\mathcal{M}} f, \nabla_{\mathcal{M}} f \rangle \qquad (5)$$

The term $\int_{\mathcal{M}} \langle \nabla_{\mathcal{M}} f, \nabla_{\mathcal{M}} f \rangle$ may be approximated on the basis of labeled and unlabeled data using the graph Laplacian ([1]). Thus, given a set of l labeled examples $\{(x_i, y_i)\}_{i=1}^{l}$ and a set of u unlabeled examples $\{x_j\}_{j=l+1}^{j=l+u}$, we consider the following optimization problem :

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A \|f\|_K^2 + \frac{\gamma_I}{(u+l)^2} \hat{f}^T L \hat{f} \qquad (6)$$

where $\hat{f} = [f(x_1), \ldots, f(x_{l+u})]^T$, and L is the graph Laplacian given by $L = D - W$ where W_{ij} are the edge weights in the data adjacency graph. Here, the diagonal matrix D is given by $D_{ii} = \sum_{j=1}^{l+u} W_{ij}$. The normalizing coefficient $\frac{1}{(u+l)^2}$ is the natural scale factor for the empirical estimate of the Laplace operator. On a sparse adjacency graph it may be replaced by $\sum_{i,j=1}^{l+u} W_{ij}$.

The following simple version of the representer theorem shows that the minimizer has an expansion in terms of both labeled and unlabeled examples and is a key to our algorithms.

Theorem 2.2. *The minimizer of optimization problem 6 admits an expansion*

$$f^*(x) = \sum_{i=1}^{l+u} \alpha_i K(x_i, x) \tag{7}$$

in terms of the labeled and unlabeled examples.

The proof is a variation of the standard orthogonality argument, which we omit for lack of space.

Remarks : (a) Other natural choices of $\| \|_I$ exist. Examples are (i) heat kernel (ii) iterated Laplacian (iii) kernels in geodesic coordinates. The above kernels are geodesic analogs of similar kernels in Euclidean space. (b) Note that K restricted to $\mathcal{M}$ (denoted by $K_{\mathcal{M}}$) is also a kernel defined on $\mathcal{M}$ with an associated RKHS $\mathcal{H}_{\mathcal{M}}$ of functions $\mathcal{M} \to \mathbb{R}$. While this might suggest $\|f\|_I = \|f|_{\mathcal{M}}\|_{K_M}$ ($f|_{\mathcal{M}}$ is f restricted to $\mathcal{M}$) as a reasonable choice for $\|f\|_I$, it turns out, that for the minimizer f^* of the corresponding optimization problem we get $\|f^*\|_I = \|f^*\|_K$, yielding the same solution as standard regularization, although with a different γ.

3 Algorithms

We now present solutions to the optimization problem posed in Eqn (6). To fix notation, we assume we have l labeled examples $\{(x_i, y_i)\}_{i=1}^{l}$ and u unlabeled examples $\{x_j\}_{j=l+1}^{j=l+u}$. We use K interchangeably to denote the kernel function or the Gram matrix.

3.1 Laplacian Regularized Least Squares (LapRLS)

The Laplacian Regularized Least Squares algorithm solves Eqn (6) with the squared loss function: $V(x_i, y_i, f) = (y_i - f(x_i))^2$. Since the solution is of the form given by (7), the objective function can be reduced to a convex differentiable function of the $(l+u)$-dimensional expansion coefficient vector $\alpha = [\alpha_1, \ldots, \alpha_{l+u}]^T$ whose minimizer is given by :

$$\alpha^* = (JK + \gamma_A lI + \frac{\gamma_I l}{(u+l)^2} LK)^{-1} Y \tag{8}$$

Here, K is the $(l+u) \times (l+u)$ Gram matrix over labeled and unlabeled points; Y is an $(l+u)$ dimensional label vector given by - $Y = [y_1, \ldots, y_l, 0, \ldots, 0]$ and J is an $(l+u) \times (l+u)$ diagonal matrix given by - $J = diag(1, \ldots, 1, 0, \ldots, 0)$ with the first l diagonal entries as 1 and the rest 0.

Note that when $\gamma_I = 0$, Eqn (8) gives zero coefficients over unlabeled data. The coefficients over labeled data are exactly those for standard RLS.

3.2 Laplacian Support Vector Machines (LapSVM)

Laplacian SVMs solve the optimization problem in Eqn. 6 with the soft margin loss function defined as $V(x_i, y_i, f) = \max(0, 1 - y_i f(x_i)), y_i \in \{-1, +1\}$. Introducing slack variables, using standard Lagrange Multiplier techniques used for deriving SVMs [16], we first arrive at the following quadratic program in l dual variables β :

$$\beta^\star = \max_{\beta \in \mathbb{R}^l} \sum_{i=1}^{l} \beta_i - \frac{1}{2} \beta^T Q \beta \tag{9}$$

subject to the contraints : $\sum_{i=1}^{l} y_i \beta_i = 0, \quad 0 \leq \beta_i \leq \frac{1}{l} \quad , i = 1, ...l$, where

$$Q = YJK(2\gamma_A I + 2\frac{\gamma_I}{(u+l)^2} LK)^{-1} J^T Y \tag{10}$$

Here, Y is the diagonal matrix $Y_{ii} = y_i$, K is the Gram matrix over both the labeled and the unlabeled data; L is the data adjacency graph Laplacian; and J is an $l \times (l+u)$ matrix given by - $J_{ij} = 1$ if $i = j$ and x_i is a labeled example, and $J_{ij} = 0$ otherwise. To obtain the optimal expansion coefficient vector $\alpha^* \in \mathbb{R}^{(l+u)}$, one has to solve the following linear system after solving the quadratic program above :

$$\alpha^* = (2\gamma_A I + 2\frac{\gamma_I}{(u+l)^2} LK)^{-1} J^T Y \beta^\star \tag{11}$$

One can note that when $\gamma_I = 0$, the SVM QP and Eqns (10,11), give zero expansion coefficients over the unlabeled data. The expansion coefficients over the labeled data and the Q matrix are as in standard SVM, in this case. Laplacian SVMs can be easily implemented using standard SVM software and packages for solving linear systems.

The Manifold Regularization algorithms and some connections are presented in the table below. For Graph Regularization and Label Propagation see [12, 3, 18].

	Manifold Regularization Algorithms
Input:	l labeled examples $\{(x_i, y_i)\}_{i=1}^{l}$, u unlabeled examples $\{x_j\}_{j=l+1}^{l+u}$
Output:	Estimated function $f : \mathbb{R}^n \to \mathbb{R}$
Step 1	▶ Construct data adjacency graph with $(l+u)$ nodes using, e.g, k nearest neighbors. Choose edge weights W_{ij}, e.g. binary weights or heat kernel weights $W_{ij} = e^{-\|x_i - x_j\|^2/4t}$.
Step 2	▶ Choose a kernel function $K(x, y)$. Compute the Gram matrix $K_{ij} = K(x_i, x_j)$.
Step 3	▶ Compute graph Laplacian matrix : $L = D - W$ where D is a diagonal matrix given by $D_{ii} = \sum_{j=1}^{l+u} W_{ij}$.
Step 4	▶ Choose γ_A and γ_I.
Step 5	▶ Compute α^* using Eqn (8) for squared loss (Laplacian RLS) or using Eqns (10,11) together with the SVM QP solver for soft margin loss (Laplacian SVM).
Step 6	▶ Output function $f^*(x) = \sum_{i=1}^{l+u} \alpha_i^* K(x_i, x)$.
	Connections to other algorithms
$\gamma_A \geq 0 \;\; \gamma_I \geq 0$	*Manifold Regularization*
$\gamma_A \geq 0 \;\; \gamma_I = 0$	*Standard Regularization (RLS or SVM)*
$\gamma_A = 0 \;\; \gamma_I > 0$	*Out-of-sample extension for Graph Regularization (RLS or SVM)*
$\gamma_A = 0 \;\; \gamma_I \to 0$	*Out-of-sample extension for Label Propagation (RLS or SVM)*
$\gamma_A \to 0 \;\; \gamma_I = 0$	*Hard margin (RLS or SVM)*

4 Related Work

In this section we survey various approaches to semi-supervised and transductive learning and highlight connections of Manifold Regularization to other algorithms.

Transductive SVM (TSVM) [16], [11]: TSVMs are based on the following optimization principle :

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}_K, y_{l+1}, \ldots y_{l+u}} C \sum_{i=0}^{l} (1 - y_i f(x_i))_+ + C^* \sum_{i=l+1}^{l+u} (1 - y_i f(x_i))_+ + \|f\|_K^2$$

which proposes a joint optimization of the SVM objective function over binary-valued labels on the unlabeled data and functions in the RKHS. Here, C, C^* are parameters that control the relative hinge-loss over labeled and unlabeled sets. The joint optimization is implemented in [11] by first using an inductive SVM to label the unlabeled data and then iteratively solving SVM quadratic programs, at each step switching labels to improve the objective function. However this procedure is susceptible to local minima and requires an unknown, possibly large number of label switches before converging. Note that even though TSVM were inspired by transductive inference, they do provide an out-of-sample extension.

Semi-Supervised SVMs (S^3VM) [5] : S^3VM incorporate unlabeled data by including the minimum hinge-loss for the two choices of labels for each unlabeled example. This is formulated as a mixed-integer program for linear SVMs in [5] and is found to be intractable for large amounts of unlabeled data. The presentation of the algorithm is restricted to the linear case.

Measure-Based Regularization [9]: The conceptual framework of this work is closest to our approach. The authors consider a gradient based regularizer that penalizes variations of the function more in high density regions and less in low density regions leading to the following optimization principle:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{i=0}^{l} V(f(x_i), y_i) + \gamma \int_X \langle \nabla f(x), \nabla f(x) \rangle p(x) dx$$

where p is the density of the marginal distribution $\mathcal{P}_X$. The authors observe that it is not straightforward to find a kernel for arbitrary densities p, whose associated RKHS norm is $\int \langle \nabla f(x), \nabla f(x) \rangle p(x) dx$. Thus, in the absence of a representer theorem, the authors propose to perform minimization over the linear space $\mathcal{F}$ generated by the span of a fixed set of basis functions chosen apriori. It is also worth noting that while [9] uses the gradient $\nabla f(x)$ in the ambient space, we use the penalty functional associated with the gradient $\nabla_{\mathcal{M}} f$ over a submanifold. In a situation where the data truly lies on or near a submanifold $\mathcal{M}$, the difference between these two penalizers can be significant since smoothness in the normal direction to the data manifold is irrelevant to classification or regression. The algorithm in [9] does not demonstrate per-

formance improvements in real world experiments.

Graph Based Approaches See e.g., [7, 10, 15, 17, 18, 1]: A variety of graph based methods have been proposed for transductive inference. However, these methods do not provide an out-of-sample extension. In [18], nearest neighbor labeling for test examples is proposed once unlabeled examples have been labeled by transductive learning. In [10], test points are approximately represented as a linear combination of training and unlabeled points in the feature space induced by the kernel. We also note the very recent work [6] on out-of-sample extensions for semi-supervised learning. For Graph Regularization and Label Propagation see [12, 3, 18]. Manifold regularization provides natural out-of-sample extensions to several graph based approaches. These connections are summarized in the Table on page 5.

Other methods with different paradigms for using unlabeled data include Cotraining [8] and Bayesian Techniques, e.g., [14].

5 Experiments

We performed experiments on a synthetic dataset and two real world classification problems arising in visual and speech recognition. Comparisons are made with inductive methods (SVM, RLS). We also compare with Transductive SVM (e.g., [11]) based on our survey of related algorithms in Section 4. For all experiments, we constructed adjacency graphs with 6 nearest neighbors. Software and Datasets for these experiments are available at **http://manifold.cs.uchicago.edu/manifold_regularization**. More detailed experimental results are presented in [4].

5.1 Synthetic Data : Two Moons Dataset

The two moons dataset is shown in Figure 2. The best decision surfaces across a wide range of parameter settings are also shown for SVM, Transductive SVM and Laplacian SVM. The dataset contains 200 examples with only 1 labeled example for each class. Figure 2 demonstrates how TSVM fails to find the optimal solution. The Laplacian SVM decision boundary seems to be intuitively most satisfying. Figure 3 shows how increasing the intrinsic regularization allows effective use of unlabeled data for constructing classifiers.

5.2 Handwritten Digit Recognition

In this set of experiments we applied Laplacian SVM and Laplacian RLSC algorithms to 45 binary classification problems that arise in pairwise classification of handwritten digits. The first 400 images for each digit in the USPS training set (preprocessed using PCA to 100 dimensions) were taken to form the training set and 2 of these were randomly labeled. The remaining images formed the test set. Polynomial Kernels of degree 3 were used, and $\gamma l = 0.05 (C = 10)$ was set for inductive methods following experiments reported in [13]. For manifold regularization, we chose to split the same weight in the ratio $1 : 9$ so that $\gamma_A l = 0.005, \frac{\gamma_I l}{(u+l)^2} = 0.045$. The observations reported in this section hold consistently across a wide choice of parameters. In Figure 4, we compare the error rates of Laplacian algorithms, SVM and TSVM, at the precision-recall breakeven points in the ROC curves (averaged over 10 random choices of labeled examples) for the 45 binary classification problems. The following comments can be made: (a) Manifold regularization results in significant improvements over inductive classification, for both RLS and SVM, and either compares well or significantly outperforms TSVM across the 45 classification problems. Note that TSVM solves multiple quadratic programs in the size of the labeled and unlabeled sets whereas LapSVM solves a single QP in the size of the labeled set, followed by a linear system. This resulted in substantially faster training times for LapSVM in this experiment. (b) Scatter plots of performance on test and unlabeled data sets confirm that the out-of-sample extension is good for both LapRLS and LapSVM. (c) Finally, we found Laplacian algorithms to be significantly more stable with respect to choice of the labeled data than the inductive methods and TSVM, as shown in the scatter plot in Figure 4 on standard deviation of error rates. In Figure 5, we plot performance as a function of number of labeled examples.

5.3 Spoken Letter Recognition

This experiment was performed on the Isolet database of letters of the English alphabet spoken in isolation (available from the UCI machine learning repository). The data set contains utterances of 150 subjects who spoke the name of each letter of the English alphabet twice. The speakers are grouped into 5 sets of 30 speakers each, referred to as isolet1 through isolet5. For the purposes of this experiment, we chose to train on the first 30 speakers (isolet1) forming a training set of 1560 examples, and test on isolet5 containing 1559 examples (1 utterance is missing in the database due to poor recording). We considered the task of classifying the first 13 letters of the English alphabet from the last 13. The experimental set-up is meant to simulate a real-world situation: we considered 30 binary classification problems corresponding to 30 splits of the training data where all 52 utterances

Figure 2: Two Moons Dataset: Best decision surfaces using RBF kernels for SVM, TSVM and Laplacian SVM. Labeled points are shown in color, other points are unlabeled.

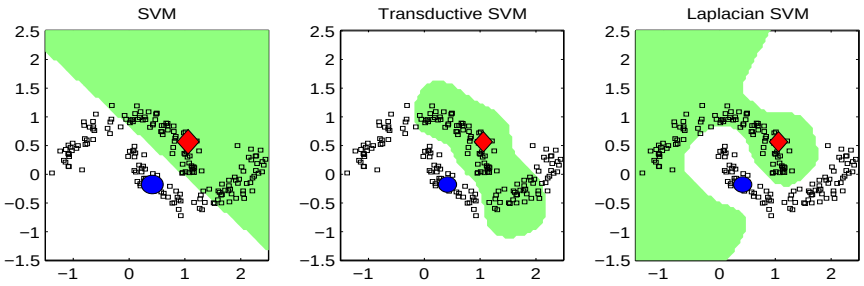


Figure 3: Two Moons Dataset: Laplacian SVM with increasing intrinsic regularization.

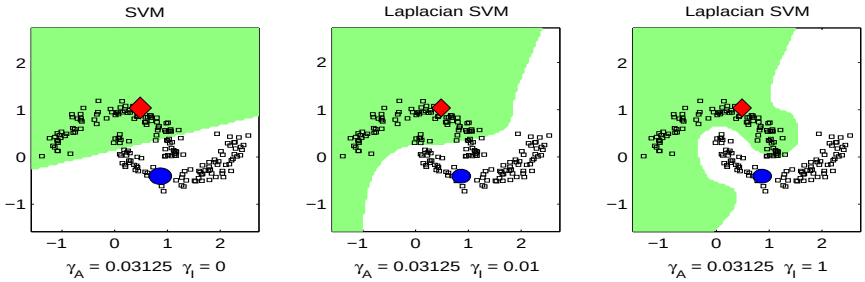


Figure 4: USPS Experiment - Error Rates at Precision-Recall Breakeven points for 45 binary classification problems

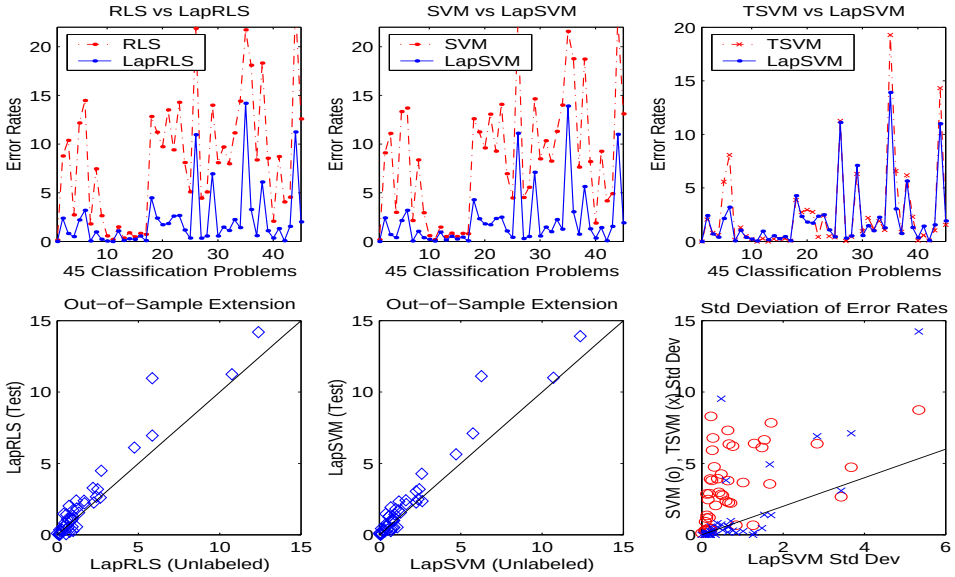


Figure 5: USPS Experiment - Mean Error Rate at Precision-Recall Breakeven points as a function of number of labeled points (T: Test Set, U: Unlabeled Set)

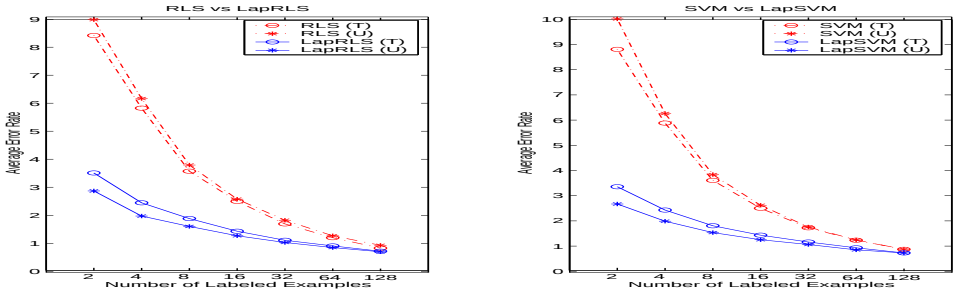
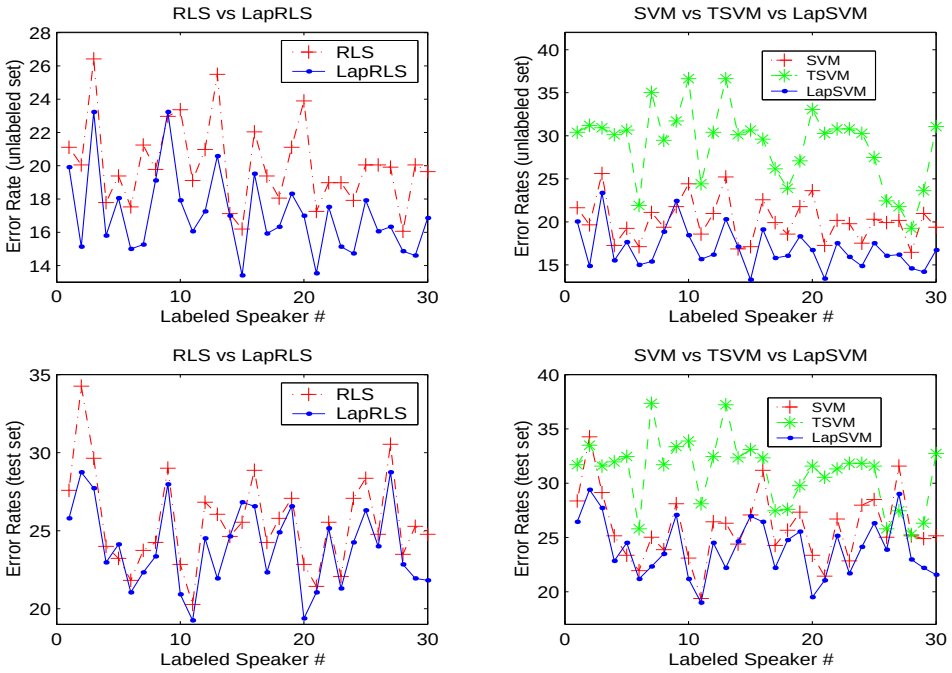


Figure 6: Isolet Experiment - Error Rates at precision-recall breakeven points of 30 binary classification problems



of one speaker were labeled and all the rest were left unlabeled. The test set is composed of entirely new speakers, forming the separate group isolet5.

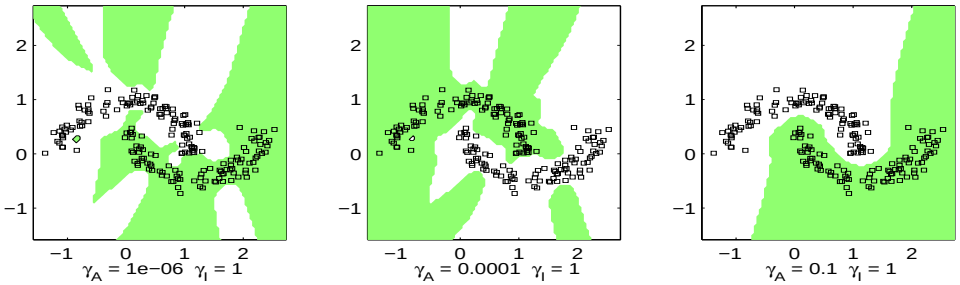
We chose to train with RBF kernels of width $\sigma = 10$ (this was the best value among several settings with respect to 5-fold cross-validation error rates for the fully supervised problem using standard SVM). For SVM and RLSC we set $\gamma l = 0.05$ ($C = 10$) (this was the best value among several settings with respect to mean error rates over the 30 splits). For Laplacian RLS and Laplacian SVM we set $\gamma_A l = \frac{\gamma_I l}{(u+l)^2} = 0.005$. In Figure 6, we compare these algorithms. The following comments can be made: (a) LapSVM and LapRLS make significant performance improvements over inductive methods and TSVM, for predictions on unlabeled speakers that come from the same group as the labeled speaker, over all choices of the labeled speaker. (b) On Isolet5 which comprises of a separate group of speakers, performance improvements are smaller but consistent over the choice of the labeled speaker. This can be expected since there appears to be a systematic bias that affects all algorithms, in favor of same-group speakers. For further details, see [4].

5.4 Regularized Spectral Clustering and Data Representation

When all training examples are unlabeled, the optimization problem of our framework, expressed in Eqn 6 reduces to the following clustering objective function :

$$\min_{f \in \mathcal{H}_K} \gamma \|f\|_K^2 + \hat{f}^T L \hat{f} \qquad (12)$$

Figure 7: Two Moons Dataset: Regularized Clustering



where $\gamma = \frac{\gamma_A u^2}{\gamma_I}$ is a regularization parameter that controls the complexity of the clustering function. To avoid degenerate solutions we need to impose some additional conditions (cf. [2]). It can be easily seen that a version of Representer theorem holds so that the minimizer has the form $f^* = \sum_{i=1}^{u} \alpha_i K(x_i, \cdot)$ By substituting back in Eqn. 12, we come up with the following optimization problem:

$$\alpha = \underset{\substack{\mathbf{1}^T K\alpha=0 \\ \alpha^T K^2 \alpha=1}}{\operatorname{argmin}} \ \gamma \|f\|_K^2 + \hat{f}^T L \hat{f}$$

where $\mathbf{1}$ is the vector of all ones and $\alpha = (\alpha_1, \ldots, \alpha_u)$ and K is the corresponding Gram matrix.

Letting P be the projection onto the subspace of $\mathbb{R}^u$ orthogonal to $K\mathbf{1}$, one obtains the solution for the constrained quadratic problem, which is given by the generalized eigenvalue problem

$$P(\gamma K + KLK)P\mathbf{v} = \lambda P K^2 P\mathbf{v} \qquad (13)$$

The final solution is given by $\alpha = P\mathbf{v}$, where $\mathbf{v}$ is the eigenvector corresponding to the smallest eigenvalue.

The framework for clustering sketched above provides a regularized form spectral clustering, where γ controls the smoothness of the resulting function in the ambient space. We also obtain a natural out-of-sample extension for clustering points not in the original data set. Figure 7 shows the results of this method on a two-dimensional clustering problem.

By taking multiple eigenvectors of the system in Eqn. 13 we obtain a natural regularized out-of-sample extension of Laplacian eigenmaps [1]. This leads to new method for dimensionality reduction and data representation. Further study of this approach is a direction of future research.

Acknowledgments. We are grateful to Marc Coram, Steve Smale and Peter Bickel for intellectual support and to NSF funding for financial support. We would like to acknowledge the Toyota Technological Institute for its support for this work.

References

[1] M. Belkin, P. Niyogi, *Using Manifold Structure for Partially Labeled Classification*, NIPS 2002.

[2] M. Belkin, P. Niyogi, *Laplacian Eigenmaps for Dimensionality Reduction and Data Representation*, Neural Computation, June 2003

[3] M. Belkin, I. Matveeva, P. Niyogi, *Regression and Regularization on Large Graphs*, COLT 2004.

[4] M. Belkin, P. Niyogi, V. Sindhwani, *Manifold Regularization : A Geometric Framework for Learning From Examples*, Technical Report, Univ. of Chicago, Department of Computer Science, TR-2004-06. Available at : http://www.cs.uchicago.edu/research/publications/techreports/TR-2004-06

[5] K. Bennett and A. Demirez, *Semi-Supervised Support Vector Machines*, NIPS 1998

[6] Y. Bengio, O. Delalleau and N.Le Roux, *Efficient Non-Parametric Function Induction in Semi-Supervised Learning*, Technical Report 1247, DIRO, University of Montreal, 2004.

[7] A. Blum, S. Chawla, *Learning from Labeled and Unlabeled Data using Graph Mincuts*, ICML 2001.

[8] A. Blum, T. Mitchell, *Combining Labeled and Unlabeled Data with Co-Training*, COLT 1998

[9] O. Bousquet, O. Chapelle, M. Hein, *Measure Based Regularization*, NIPS 2003

[10] Chapelle, O., J. Weston and B. Schoelkopf, *Cluster Kernels for Semi-Supervised Learning*, NIPS 2002.

[11] T. Joachims, *Transductive Inference for Text Classification using Support Vector Machines*, ICML 1999.

[12] A. Smola and R. Kondor, *Kernels and Regularization on Graphs*, COLT/KW 2003.

[13] B. Schoelkopf, C.J.C. Burges, V. Vapnik, *Extracting Support Data for a Given Task*, KDD95.

[14] M. Seeger *Learning with Labeled and Unlabeled Data*, Tech Report. Edinburgh University (2000)

[15] Martin Szummer, Tommi Jaakkola, *Partially labeled classification with Markov random walks*, NIPS 2001,.

[16] V. Vapnik, *Statistical Learning Theory*, Wiley-Interscience, 1998.

[17] D. Zhou, O. Bousquet, T.N. Lal, J. Weston and B. Schoelkopf, *Learning with Local and Global Consistency*, NIPS 2003.

[18] X. Zhu, J. Lafferty and Z. Ghahramani, *Semi-supervised learning using Gaussian fields and harmonic functions*, ICML 2003.

Distributed Latent Variable Models of Lexical Co-occurrences

John Blitzer
Department of Computer
and Information Science
University of Pennsylvania
Philadelphia, PA 19104

Amir Globerson
Interdisciplinary Center
for Neural Computation
The Hebrew University
Jerusalem 91904, Israel

Fernando Pereira
Department of Computer
and Information Science
University of Pennsylvania
Philadelphia, PA 19104

Abstract

Low-dimensional representations for lexical co-occurrence data have become increasingly important in alleviating the sparse data problem inherent in natural language processing tasks. This work presents a distributed latent variable model for inducing these low-dimensional representations. The model takes inspiration from both connectionist language models [1, 16] and latent variable models [13, 9]. We give results which show that the new model significantly improves both bigram and trigram models.

1 Introduction

Data sparseness is a central problem in machine learning for natural-language processing tasks such as parsing, language modeling, machine translation, and automatic summarization [5, 10, 3]. All these tasks use sparse data to estimate lexical co-occurrence probabilities. One example is modeling the probability $p(w|h)$ that a word w occurs in the context of some other word or words h.

The most common way of dealing with the sparse data problem is to interpolate the full context with shorter contexts, combining the full and shortened statistics into a smoother overall estimate (see [4]). Interpolation models are simple and reasonably effective, but they fail to take into account useful regularities across contexts.

An effective approach to capturing those regularities is to map contexts into low-dimensional representations, and then use those representations to predict w. Two lines of work which have used these ideas are latent variable models and distributed connectionist models.

In latent variable models, one introduces an unobserved discrete random variable with low cardinality, which represents an underlying class of contexts, and separates the context from the word w. The parameters of the model are then estimated via EM or related algorithms [13, 9].

Distributed models map contexts to continuous low dimensional vectors $g(h) \in \Re^n$. The distribution $p(w|h)$ is then modeled as a non linear function of $g(h)$. Recent work by Bengio *et al.* [1] and by Xu *et al.* [16] has shown that this approach yields state of the art performance.

The multidimensional representation of distributed models is attractive, since it can be thought of as representing different, possibly independent, properties of the context. On the other hand, in latent models the intermediate representation has a clear probabilistic interpretation.

In this work, we suggest a model that combines the advantages of the above approaches by using a *vector* of discrete latent variables. Our latent variable model can be nicely described as a directed graphical model, which also encompasses single latent variable models such as the aggregate Markov model (AMM) of Saul and Pereira [13]. Because it uses a vector of latent variables, it also extends naturally to more general n-grams. We show empirically that our model significantly outperforms the AMM on a verb-object bigram modeling task, and we further demonstrate improved performance over a standard baseline for trigram lan-

guage modeling.

The rest of the paper is organized as follows. Section 2 reviews the AMM and introduces the distributed latent variable model. Section 3 describes the EM algorithms for optimizing the model. Section 4 describes our experimental procedures and results. We conclude in Section 6.

2 Distributed Latent Variable Models

In what follows, we describe a model for the probability of observing a word w given a history of previous words h. We begin with a single word history, in order to compare to standard latent models, and then discuss extensions to multivariate histories. The simplest, *non-distributed*, latent model assumes the existence of a low cardinality hidden variable z which separates w and h in the sense that it makes them conditionally independent

$$p(w, h|z) = p(w|z)p(h|z) \qquad .$$

The corresponding graphical model is the Bayes net in Figure 1. The conditional probability of w given h is obtained by marginalizing over z:

$$p(w|h) = \sum_z p(z|h) \cdot p(w|z) \qquad .$$

The parameters of this model are $p(z|h)$ and $p(w|z)$. The expression for the conditional distribution suggests that z can be thought of as a clustering of h, from which w is then predicted.

Saul and Pereira [13] used this model for language modeling, where they called it the aggregate Markov model (AMM). Hofmann and Puzicha [9] studied it as a special case of what they call "aspect" models. The AMM also falls under the rubric of mixture models. The distribution $p(w|h)$ can be thought of as a mixture of $|z|$ low-dimensional distributions, $p(w|z)$, each of which has mixing weight $p(z|h)$. We will make use of this view of the latent variable model in our analysis.

To use the advantages of a distributed model, we now assume that the latent variable is a *vector* of m binary latent variables (bits)[1] $\mathbf{b} = (b_1, \ldots, b_m)$. As in the previous latent variable model, the latent variables are assumed to separate h and w.

[1]Using binary latent variables simplifies the notation, exposition, and implementation without real loss of generality.

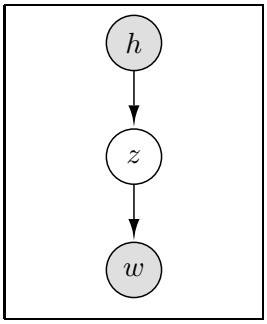


Figure 1: AMM graphical model

A desired property of a distributed model is that its vector elements model independent properties of their inputs. This can be achieved in our model by constraining the variables b_i to be conditionally independent given h. The above requirement can be represented graphically using the Bayes net of Figure 2. To further exploit the distributed nature of $\mathbf{b}$, we use the following multinomial logistic (also known as conditional maximum entropy) model for $p(w|\mathbf{b})$

$$p(w|\mathbf{b}) = \frac{1}{Z(\mathbf{b})} \exp \sum_{i=1}^{m} \psi(b_i, w) \qquad (1)$$

with separate interaction potentials $\psi(b_i, w)$ for each variable b_i. As in standard condtional maximum entropy models, we assume $\psi(b_i, w)$ is an arbitrary, real-valued function of b_i and w.

We refer to the above model as the distributed Markov model (henceforth DMM). The distribution it induces is specified as follows:

$$\begin{aligned} p(w|h) &= \sum_{\mathbf{b}} p(\mathbf{b}|h) \cdot p(w|\mathbf{b}) \\ &= \sum_{\mathbf{b}} \prod_{i=1}^{m} p(b_i|h) \cdot \frac{1}{Z(\mathbf{b})} \exp \sum_{i=1}^{m} \psi(b_i, w) \qquad . \end{aligned}$$

The parameters of the model are the binomial parameters $p(b_i|h)$, and the real-valued functions $\psi(b_i, w)$.

The distribution $p(w|h)$ can be seen to be a mixture of 2^m components. However, the DMM has only $O(m)$ parameters. This should be contrasted with the AMM which needs $O(2^m)$ parameters to model a mixture of the same size. When the latent variable has a distributed nature, we expect the DMM model to outperform the standard mixture model, since its fewer pa-

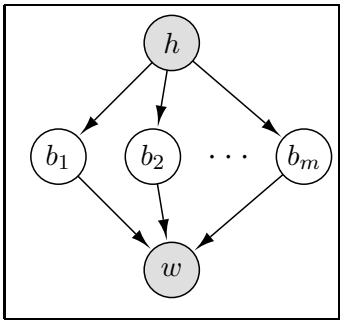


Figure 2: Graphical model corresponding to the distributed latent variable model.

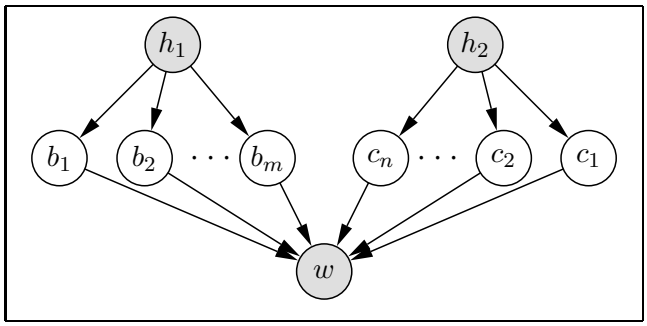


Figure 3: Graphical model corresponding to the multivariate history DMM

rameters make it less likely to overfit. This is indeed seen in the experiments described later.

2.1 Multivariate Histories

The distributed model is easily extended to multi-word histories. For ease of exposition, we discuss here the two-word history (trigram) case. Each word in the history is encoded with a separate vector of latent variables as shown in the graphical model of of Figure 3.

We denote latent variables encoding h_1 and h_2 by $\mathbf{b} = (b_1, \ldots, b_m)$ and $\mathbf{c} = (c_1, \ldots, c_n)$ respectively. The distribution $p(w|\mathbf{b}, \mathbf{c})$ is assumed to factor with respect to the latent variables as in Equation 1. The interaction potentials for $\mathbf{c}$ will be denoted by $\psi(c_i, w)$. The resulting distribution $p(w|h_1, h_2)$ is a mixture of 2^{m+n} elements, with $O(m+n)$ parameters. The vectors $\mathbf{b}$ and $\mathbf{c}$ might have different lengths, allowing us to use more latent variables for more recent history elements, for example.

3 Learning Distributed Models

We now discuss the estimation of the parameters specifying the distributed model, from a given training set. We focus on the two-word history case. Since the model is a mixture distribution, its parameters may be estimated using the expectation-maximization algorithm [7]. Applications of EM to learning single latent variable models are described by Saul and Pereira [13], and Hofmann and Puzicha [9].

In what follows we denote by $\Theta^{(t)}$ the full parameter set at the iteration t of the EM algorithm. Recall that the free parameters are the binomial parameters $p(b_i|h_1), p(c_j|h_2)$ and the functions $\psi(b_i, w), \psi(c_j, w)$. We denote all parameters at time t with a superscript t.

In the E step of the algorithm, posterior probabilities of the latent variables are calculated given the model $\Theta^{(t)}$. For the DMM, this posterior is [2]

$$p^{(t+1)}(\mathbf{b}, \mathbf{c}|w, h, \Theta^{(t)}) \propto p^{(t)}(w|\mathbf{b}, \mathbf{c}) p^{(t)}(\mathbf{b}|h_1) p^{(t)}(\mathbf{c}|h_2)$$

Observe that, because of the DMM's factored form, the posterior can be computed as a normalized product of mixture weights and mixture components. Because of this, the histories h and predicted words w decouple, and we can compute the partition function $Z(\mathbf{b}, \mathbf{c})$ independently of h. Thus the E step for the model requires $O(L \cdot 2^{m+n})$ time, where L refers to the total number of training instances. This can be significantly smaller than the required time for other distributed models.

From this, we construct the joint distribution $p^{(t+1)}(h, w, \mathbf{b}, \mathbf{c})$ over the observed and latent variables

$$p^{(t+1)}(h, w, \mathbf{b}, \mathbf{c}) \propto p^{(t+1)}(\mathbf{b}, \mathbf{c}|h, w, \Theta^{(t)}) p_0(h, w)$$

where $p_0(h, w)$ is the empirical distribution of the observed variables.

The M step uses the above posteriors to estimate a new parameter set $\Theta^{(t+1)}$. The parameters $p^{(t+1)}(b_i|h_1)$ and $p^{(t+1)}(c_j|h_2)$ are easily obtained by marginalization and conditionalization from $p^{(t+1)}(h, w, \mathbf{b}, \mathbf{c})$.

The parameters $\psi(b_i, w), \psi(c_j, w)$ have no closed-form expression, due to the dependencies between

[2] We use h to denote (h_1, h_2) for brevity

the latent variables induced by the partition function $Z(\mathbf{b}, \mathbf{c})$. However, as is standard for conditional maximum entropy models like that of Equation 1, the expected complete data log-likelihood is maximized for the values of the parameter vector $\psi^{(t+1)}$ that satisfy the marginal constraints

$$\begin{aligned} q(w, b_i) &= p^{(t+1)}(w, b_i) \\ q(w, c_j) &= p^{(t+1)}(w, c_i) \end{aligned}$$

where

$$q(w, \mathbf{b}, \mathbf{c}) = p(w|\mathbf{b}, \mathbf{c}, \psi^{(t+1)})p(\mathbf{b}, \mathbf{c})$$

is the model-expected marginal count of w and $\mathbf{b}, \mathbf{c}$. The above equations can be solved iteratively for $\psi^{(t+1)}(w, b_i)$ with generalized iterative scaling (GIS) [6], which uses the following parameter update

$$\Delta\psi^{(t+1)}(w, b_i) = \frac{1}{C} \log \frac{p^{(t+1)}(w, b_i)}{q(w, b_i)}$$

where C is an upper bound on the number of terms in the exponent in Equation 1. Similar updates are used for $\psi(w, c_i)$.

3.1 Overrelaxed Updates and Regularization

In this section we discuss briefly two practical issues related to the learning algorithm.

First, both GIS and EM are known to be slow to converge, so that the doubly-iterative EM algorithm proposed in the previous section is impractically slow without modification. To make the algorithm practical, we use only a few GIS steps per EM step and then apply an adaptive, overrelaxed update [12]. For large models, this modified update gives convergence in under 75 iterations, whereas the standard EM update does not.

Secondly, while both latent variable models are intended to smooth a conditional distribution, they can also overfit their training data, in particular when the number of mixture components is large. Because of this, we sometimes use a deterministic annealing variant of EM [9, 14], where a free parameter corresponding to temperature is optimized on heldout data.

4 Experimental Evaluations

To evaluate our distributed latent variable model we first apply it to modeling bigrams, in order to compare it to single latent variable models. We then apply it to the more complex task of trigram modeling.

4.1 Modeling Verb Object Association

We begin with modeling the co-occurrence of bigrams composed of verbs followed by objects. The task is to obtain a model of the conditional distribution $p(o|v)$ of seeing an object given the preceding verb.

Our data come from a subset of the Penn Treebank Wall Street Journal Corpus [11]. This is a set of one million words of Wall Street Journal text, where each sentence is annotated with its syntactic structure in the form of a parse tree. The data are extracted from the parse trees by finding all right branching parent-modifier pairs where the parent is tagged as a verb and the child is tagged as a noun. Instances of the data then correspond roughly to verbs and objects, both direct and indirect, but also can include other nouns such as predicate nominals and the subjects in subject-verb inversions. The data contain 50,205 such verb-object pairs, and there are 5191 unique verbs and 8470 unique nouns, so the matrix is quite sparse. In order to avoid the problem of unseen words, we collapsed all low-count words into a single token. The count threshold for this normalization also affects the sparseness of the matrix, and we performed experiments which varied it.

The dataset was divided into 9-1 training-test splits, and we further divided the training set to give us held-out data on which to optimize model free parameters.

We use a bigram model smoothed by deleted interpolation as a baseline. This model gives the probability of an object given a verb as:

$$p_{BG}^{interp}(o|v) = \lambda_1 p_{ML}(o|v) + \lambda_2 p_{ML}(o) + \lambda_3 p_U \quad (2)$$

Here the weights λ_i are positive and sum to 1. The three distributions interpolated are the maximum likelihood bigram, unigram, and uniform distributions. Following Jelinek [10], we bin the histories based on frequency and set the weights to maximize the likelihood of heldout data.

Given a distributed model $p_{DM}(o|v)$ we interpolate it with the baseline by adding the term $\lambda_4 p_{DM}(o|v)$ to Equation 2 and determining the weights as above.

The baseline and distributed models are evaluated using data predictive perplexity. For a model $p(o|v)$,

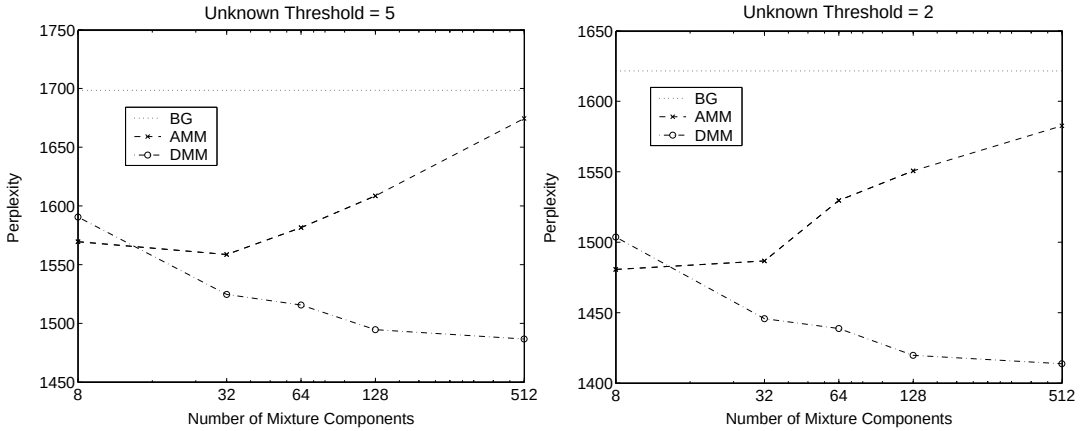


Figure 4: Perplexity Results for all three latent variable models for varying numbers of mixture components with two unknown word thresholds. An AMM with k mixture components has a latent class variable that takes on k values. A DMM with k mixture components has $log(k)$ binary latent variables

Table 1: Percentage improvement over the baseline smoothed bigram model for the best latent variable models of each type. UNK refers to the unknown word threshold.

% Improve over BG	UNK=2	UNK=5
AMM	8.6%	7.6%
DMM	12.8%	12.5%

the perplexity on a test set of verbs and objects is $\mathcal{P}_{\text{test}} = \exp\Big[-\frac{1}{n}\sum_{v,o} c(o,v)\log p(o|v)\Big]$.

Here $c(o,v)$ refers to the count of a verb-object bigram, and n is the total number of bigrams in the corpus.

Figure 4 gives perplexity results for the latent variable models with varying numbers of mixture components. The latent variable models are trained with annealed EM as described in Section 3.1, and the perplexities reported are for interpolated models as described above.

While the single latent variable model (AMM) does give better results for 8 mixture components, the distributed model consistently outperforms it for all other numbers of mixture components. It is also worthwhile to note that the AMM overfits its training data, even when annealed, while the distributed model shows no sign of overfitting, even when the number of mixture components becomes quite large. This is because the distributed model has a much more compact representation for the same number of mixture components. As mentioned before, the number of parameters in the distributed model scales logarithmically in the number of mixture components, but it scales linearly for the AMM.

As Table 1 shows, the best distributed models can improve the baseline by nearly 13%. In the next section, we examine the kinds of lexical "clusterings" discovered by the AMM and DMM.

4.1.1 Analysis of Lexical Co-occurrence Clusterings

One way to analyze the latent variable models is to examine the distributions $p(z|v)$, $p(o|z)$ for the AMM and $p(\mathbf{b}|v)$, $p(o|\mathbf{b})$ for the DMM. In the AMM, verbs v for which $p(\hat{z}|v)$ is high can be thought of as belonging to the $\hat{z}$ cluster. Similarly, objects o for which $p(o|\hat{z})$ is high can be thought of as belonging to the $\hat{z}$ cluster. Since the latent variable models here are mixture models, analyzing verbs and objects as belonging to one single cluster can hide some subtleties. Nonetheless, examining the parameters in this way provides us with useful insight into their representations of lexical co-occurrences

Table 2 shows one such "clustering" of the 200 most frequent verbs and 200 most frequent objects induced by the 32-class AMM on the verb-object data. For verbs and objects v and o, we assign them to the cluster $\hat{z}$ corresponding to $\hat{z} = \operatorname{argmax}_z p(z|v)$ and $\hat{z} = \operatorname{argmax}_z p(o|z)$ respectively. We display the top 8 clusters, sorted by the highest $p(z|v)$ value for verbs in the cluster.

Class	$p(z\|v)$	$p(o\|z)$
1	named	chairman, president
2	been, become, told, asked	issue
3	put, build	funds, system
4	did, do	it, business, something
5	are, used	issues
6	called, find, know	what
7	consider, approved, following	plan, bid, program, changes, proposal, bill
8	give, gives, giving, given, lost	support, right, money

Table 2: 8 classes induced by the 32 class AMM. The first column shows the verbs for which this class is the most probable. The second column shows the nouns which are most probable given this class.

Class	$p(\mathbf{b}\|v)$	$p(o\|\mathbf{b})$
1	said, says, were	executive
2	held, dropped, yield, rose, grew	point, %, cents, points
3	ended, ending, began, sent, reached	Nov., report, March, agreement, June
4	is, be, been, named, become	director, chairman, president, part, reason
5	closed	year, Tuesday, week, Wednesday, yesterday
6	set, consider, begin, rejected, made	acquisition, sale, plan, plans
7	told, asked, help, called, tell	us, investors, people, him, me
8	did, does, do, play, call, know	you, what, work, role, something

Table 3: 8 classes induced by the 5 variable DDM. The first column shows the verbs for which this class is the most probable. The second column shows the nouns which are most probable given this class.

Table 3 gives a similar clustering induced by the 5-variable DMM. The 8 clusters are chosen in the same fashion. Both models find some reasonable clusters of verbs and objects, but the AMM clusters are a little less sharply defined, and it induces some clusters that seem quite counterintuitive. The word "issue" is in a separate cluster from "issues," for instance.

4.2 Trigram Language Modeling

In this section, we evaluate our distributed latent variable model on trigram language modeling. Trigram models are widely used in speech recognition because they are easy to build and efficient to apply. Trigram language models also usually generalize in a straightforward way to longer contexts. Our data for this experiment also come from the treebank portion of the Wall Street Journal restricted to 10,000 distinct words. The dataset is described more completely in [17] and contains sections for training, optimizing free parameters parameters and testing. For these experiments, we do not anneal either model. Instead we regularize the models by stopping the EM algorithm when the models begin to overfit development data. For this data, annealing is not effective at regularizing the AMM, and annealing the DMM did not make a significant difference after interpolation.

Figure 5(a) gives perplexity results for distributed models with varying numbers of latent variables per word. It is natural to suppose that w_2 is the more effective predictor of w_3, and this is indeed shown in these experiments. In fact for all but the 10-bit case, it is always better to allocate all of them to the most recent word in the history. Even then, the models show little sign of saturating the amount of information present in either of the histories.

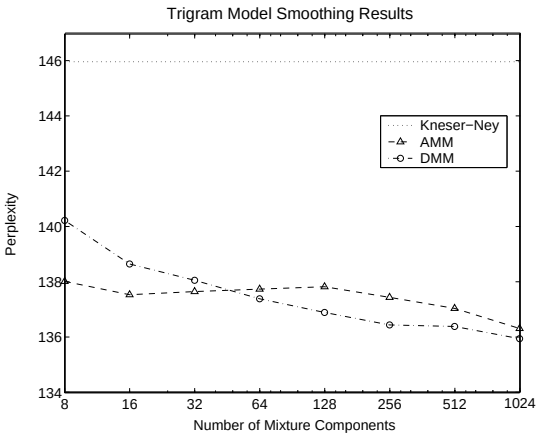
We also compared our model with an interpolated Kneser-Ney trigram model. The interpolated Kneser-Ney model is one of the most effective trigram language models, and Chen and Goodman demonstrate that a variant of this model consistently outperforms all other interpolation and backoff smoothing techniques [4] [3]. We use the Good-Turing estimate of the parameters of the Kneser-Ney model and we combine the AMM and DMM with it by linearly interpolating the two trained models. As before, we set the weights to maximize the likelihood of heldout data.

Figure 5(c) shows learning curves for the distributed model when interpolated with the trigram baseline described above. Again we compare with an aggregate Markov model. This time, the AMM is used only to smooth the distribution $p(w_3|w_2)$. As with the verb-object bigram results, we compare two models

[3] This variant, modified Kneser-Ney, can achieve a perplexity of 143.9 on this dataset.

(a) Perplexities of models with different numbers of latent variables

	$\mathbf{c}(w_1)$				
$\mathbf{b}(w_2)$	0	1	2	3	4
3	379	361	346	343	344
4	336	329	317	321	316
5	315	310	300	299	296
6	295	292	285	283	X
7	283	280	277	273	X
8	274	273	270	X	X
9	269	**261**	X	X	X
10	263	X	X	X	X



(c) Learning curves for DMM and AMM models with varying numbers of mixture components

Figure 5: Results of trigram language modeling experiments

with identical numbers of mixture components. For each bit count, we choose the DMM with the lowest heldout data perplexity. For all but 10 bits (1024 mixture components), this corresponds to allocating all bits to the most recent word. While the DMM slightly outperforms the AMM for larger numbers of mixture components, both models improve similarly (6.6 versus 6.8%) on the Kneser-Ney baseline. One explanation for the difference between these results and the verb-object bigram results is the difference in data sparseness. The AMM quickly overfits the verb-object data, but it is much harder to overfit the 10,000-word vocabulary trigram data.

5 Related Work and Discussion

Aside from Saul and Pereira [13], there is also other work on applying ideas similar to ours to language modeling. Wang et al. [15] investigate a language model based on what they call the latent maximum entropy principle. Like our distributed models, latent maximum entropy models are mixture models. Similarly, like our factored form from Equation 1, latent maximum entropy models are maximum entropy for a fixed posterior distribution over their hidden features. However, unlike our model, the Wang et al. [15] model is not distributed and uses a single latent variable.

The models which are most similar in spirit to ours are the connectionist models of Bengio [1] and Xu [17]. These models combine a neural network with a softmax output for probabilities. They are not sparse, and because of this, they must use $O(H \cdot V)$ time per epoch, where H refers to the number of unique histories and V refers to the vocabulary size. For most corpora, this number is prohibitively large. Finally, we recently introduced a much faster model which combines dimensionality reduction for continuous embedding of histories with a hierarchical mixture of experts for faster normalization [2].

Our distributed latent variable model is fully probabilistic, and its factored form allows it to avoid the prohibitively large $O(H \cdot V)$ calculation of the earlier connectionist models. Recall that the DMM requires time $O(L \cdot 2^{m+n})$. Since L is typically close to H (for our trigram data, L is 1 million and H is 600,000), this results in much faster training times when m and n are small. When m and n are large, though, the DMM also faces speed issues. The coupling of the latent variables forces us to sum over all mixture components when computing the expectations in the E-step. While this is possible for 1024-component mixtures, it becomes too large for more mixture components on larger corpora.

In terms of model form and inference, our model is closely related to the factorial hidden Markov model [8]. The factors of the factorial HMM correspond to our latent bits, but there are two major differences between the two models: The factorial HMM is a generative model in which the observed variables at a time slice are generated from the factors. By contrast, our DMM assumes that the the latent bits are distributed conditioned on a history. Furthermore, the factorial

HMM explicitly models the dependencies between hidden variables at different time slices, whereas we do not. Ghahramani and Jordan give a mean-field approximation to the posterior for their factorial HMM. Unfortunately, the introduction of the mean field parameters in our model recouples h and w, resulting in exactly the $O(H \cdot V)$ computation we avoided by specifying its factored form.

6 Conclusion

We presented a distributed latent variable model for lexical co-occurrence data. Using the framework of graphical models, we derived an EM algorithm and compared the learned models with the single latent variable aggregate Markov model [13]. The result was that the distributed latent variable model significantly outperforms the single latent variable AMM and is able to improve significantly on both bigram and trigram baselines.

As mentioned in the previous section, our model requires time exponential in the number of latent variables in order to perform exact inference. For 9 binary latent variables, this is not large, but for models with many variables, this number quickly becomes too large. This is due to the directed nature of our graphical model. It is worth mentioning that it is also possible to use undirected graphical models to represent the essential conditional independences expressed in this model. We are currently investigating such models together with approximate inference techniques [18] to make learning and inference practical.

Acknowledgements

We would like to thank Lawrence Saul and Naftali Tishby for valuable discussions throughout the course of this work. We thank Peng Xu for help preprocessing the data for the trigram language modeling experiments. Blitzer and Pereira are partially supported by DARPA under SRI subcontract NBCHD030010.

References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[2] J. Blitzer, K. Weinberger, L. Saul, and F. Pereira. Hierarchical distributed representations for statistical language modeling. In *Advances in Neural Information Processing Systems 17*, 2004.

[3] P. Brown, S. A. D. Pietra, V. J. D. Pietra, and R. L. Mercer. The mathematics of statistical machine translation. *Computational Linguistics*, 19(2):263–311, 1993.

[4] S. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of ACL*, 1996.

[5] M. Collins. *Head-driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

[6] J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematic Statistics*, 43:1470–1480, 1972.

[7] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

[8] Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. *Mach. Learn.*, 29(2-3):245–273, 1997.

[9] T. Hofmann and J. Puzicha. Statistical models for co-occurrence data. Technical Report AIM-1625, MIT, 1998.

[10] F. Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, 1997.

[11] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1994.

[12] R. Salakhutdinov and S. Roweis. Adaptive overrelaxed bound optimization methods. In *Proceedings of ICML*, 2003.

[13] L. Saul and F. Pereira. Aggregate and mixed-order Markov models for statistical language processing. In *Proceedings of EMNLP*, 1997.

[14] N. Ueda and R. Nakano. Deterministic annealing variant of the EM algorithm. In *Advances in Neural Information Processing Systems 7*, 1995.

[15] S. Wang, D. Schuurmans, F. Peng, and Y. Zhao. Semantic n-gram language modeling with the latent maximum entropy principle. In *Proceedings of AISTATS*, 2003.

[16] P. Xu, A. Emami, and F. Jelinek. Training connectionist models for the structured language model. In *Proceedings of EMNLP*, 2003.

[17] P. Xu and F. Jelinek. Random forests in language modeling. In *Proceedings of EMNLP*, 2004.

[18] J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In *IJCAI 2001 Distinguished Lecture track*.

On Contrastive Divergence Learning

Miguel Á. Carreira-Perpiñán* **Geoffrey E. Hinton**
Dept. of Computer Science, University of Toronto
6 King's College Road. Toronto, ON M5S 3H5, Canada
Email: {miguel,hinton}@cs.toronto.edu

Abstract

Maximum-likelihood (ML) learning of Markov random fields is challenging because it requires estimates of averages that have an exponential number of terms. Markov chain Monte Carlo methods typically take a long time to converge on unbiased estimates, but Hinton (2002) showed that if the Markov chain is only run for a few steps, the learning can still work well and it approximately minimizes a different function called "contrastive divergence" (CD). CD learning has been successfully applied to various types of random fields. Here, we study the properties of CD learning and show that it provides biased estimates in general, but that the bias is typically very small. Fast CD learning can therefore be used to get close to an ML solution and slow ML learning can then be used to fine-tune the CD solution.

Consider a probability distribution over a vector $\mathbf{x}$ (assumed discrete w.l.o.g.) and with parameters $\mathbf{W}$

$$p(\mathbf{x};\mathbf{W}) = \frac{1}{Z(\mathbf{W})} e^{-E(\mathbf{x};\mathbf{W})} \tag{1}$$

where $Z(\mathbf{W}) = \sum_{\mathbf{x}} e^{-E(\mathbf{x};\mathbf{W})}$ is a normalisation constant and $E(\mathbf{x};\mathbf{W})$ is an energy function. This class of random-field distributions has found many practical applications (Li, 2001; Winkler, 2002; Teh et al., 2003; He et al., 2004). Maximum-likelihood (ML) learning of the parameters $\mathbf{W}$ given an iid sample $\mathcal{X} = \{\mathbf{x}_n\}_{n=1}^N$ can be done by gradient ascent:

$$\mathbf{W}^{(\tau+1)} = \mathbf{W}^{(\tau)} + \eta \left.\frac{\partial L(\mathbf{W};\mathcal{X})}{\partial \mathbf{W}}\right|_{\mathbf{W}^{(\tau)}}$$

where the learning rate η need not be constant. The average log-likelihood is:

$$\begin{aligned} L(\mathbf{W};\mathcal{X}) &= \tfrac{1}{N}\textstyle\sum_{n=1}^N \log p(\mathbf{x}_n;\mathbf{W}) = \langle \log p(\mathbf{x};\mathbf{W})\rangle_0 \\ &= -\langle E(\mathbf{x};\mathbf{W})\rangle_0 - \log Z(\mathbf{W}) \end{aligned}$$

where $\langle\cdot\rangle_0$ denotes an average w.r.t. the data distribution $p_0(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^N \delta(\mathbf{x}-\mathbf{x}_n)$. A well-known difficulty arises in the computation of the gradient

$$\frac{\partial L(\mathbf{W};\mathcal{X})}{\partial \mathbf{W}} = -\left\langle \frac{\partial E(\mathbf{x};\mathbf{W})}{\partial \mathbf{W}} \right\rangle_0 + \left\langle \frac{\partial E(\mathbf{x};\mathbf{W})}{\partial \mathbf{W}} \right\rangle_\infty$$

where $\langle\cdot\rangle_\infty$ denotes an average with respect to the model distribution $p_\infty(\mathbf{x};\mathbf{W}) = p(\mathbf{x};\mathbf{W})$. The average $\langle\cdot\rangle_0$ is readily computed using the sample data $\mathcal{X}$, but the average $\langle\cdot\rangle_\infty$ involves the normalisation constant $Z(\mathbf{W})$, which cannot generally be computed efficiently (being a sum of an exponential number of terms). The standard approach is to approximate the average over the distribution with an average over a sample from $p(\mathbf{x};\mathbf{W})$, obtained by setting up a Markov chain that converges to $p(\mathbf{x};\mathbf{W})$ and running the chain to equilibrium (for reviews, see Neal, 1993; Gilks et al., 1996). This Markov chain Monte Carlo (MCMC) approach has the advantage of being readily applicable to many classes of distribution $p(\mathbf{x};\mathbf{W})$. However, it is typically very slow, since running the Markov chain to equilibrium can require a very large number of steps, and no foolproof method exists to determine whether equilibrium has been reached. A further disadvantage is the large variance of the estimated gradient.

To avoid the difficulty in computing the log-likelihood gradient, Hinton (2002) proposed the contrastive divergence (CD) method which approximately follows the gradient of a different function. ML learning minimises the Kullback-Leibler divergence

$$\mathrm{KL}\left(p_0 \| p_\infty\right) = \sum_{\mathbf{x}} p_0(\mathbf{x}) \log \frac{p_0(\mathbf{x})}{p(\mathbf{x};\mathbf{W})}.$$

CD learning approximately follows the gradient of the

*Current address: Dept. of Computer Science & Electrical Eng., OGI School of Science & Engineering, Oregon Health & Science University. Email: miguel@cse.ogi.edu.

difference of two divergences (Hinton, 2002):

$$\mathrm{CD}_n = \mathrm{KL}\left(p_0 \| p_\infty\right) - \mathrm{KL}\left(p_n \| p_\infty\right).$$

In CD learning, we start the Markov chain at the data distribution p_0 and run the chain for a small number n of steps (e.g. $n = 1$). This greatly reduces both the computation per gradient step and the variance of the estimated gradient, and experiments show that it results in good parameter estimates (Hinton, 2002). CD has been applied effectively to various problems (Chen and Murray, 2003; Teh et al., 2003; He et al., 2004), using Gibbs sampling or hybrid Monte Carlo as the transition operator for the Markov chain. However, it is hard to know how good the parameter estimates really are, since no comparison was done with the real ML estimates (which are impractical to compute). There has been a little theoretical investigation of the properties of contrastive divergence (MacKay, 2001; Williams and Agakov, 2002; Yuille, 2004), but important questions remain unanswered: Does it converge? If so how fast, and how are its convergence points related to the true ML estimates?

In this paper we provide some theoretical and empirical evidence that contrastive divergence can, in fact, be the basis for a very effective approach for learning random fields. We concentrate on Boltzmann machines, though our results should be more generally valid. First, we show that CD provides biased estimates in general: for almost all data distributions, the fixed points of CD are not fixed points of ML, and vice versa (section 2). We then show, by comparing CD and ML in empirical tests, that this bias is small (sections 3–4) and that an effective approach is to use CD to perform most of the learning followed by a short run of ML to clean up the solution (section 5).

To eliminate sampling noise from our investigations, we use fairly small models (with e.g. 48 parameters) for which we can compute the exact model distribution and the exact distribution at each step of the Markov chain at each stage of learning. Throughout, we take ML learning to mean exact ML learning (i.e., with $n \to \infty$ in the Markov chain) and CD_n learning to mean learning using the exact distribution of the Markov chain after n steps. The sampling noise in real MCMC estimates would create an additional large advantage that favours CD over ML learning, because CD has much lower variance in its gradient estimates.

1 ML and CD learning for two types of Boltzmann machine

We will concentrate on two types of Boltzmann machine, which are a particular case of the model of eq. (1). In Boltzmann machines, there are v visible units $\mathbf{x} = (x_1, \ldots, x_v)^T$ that encode a data vector, and h hidden units $\mathbf{y} = (y_1, \ldots, y_h)^T$; all units are binary and take values in $\{0, 1\}$.

Fully visible Boltzmann machines have $h = 0$ and the visible units are connected to each other. The energy is then $E(\mathbf{x}; \mathbf{W}) = -\frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x}$, where $\mathbf{W} = (w_{ij})$ is a symmetric $v \times v$ matrix of real-valued weights (for simplicity, we do not consider biases). We denote such a machine by VBM(v). In VBMs, the log-likelihood has a unique optimum because its Hessian is negative definite. Since $\partial E/\partial w_{ij} = -x_i x_j$, ML learning takes the form

$$w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} + \eta\left(\langle x_i x_j\rangle_0 - \langle x_i x_j\rangle_\infty\right)$$

while CD_n learning takes the form

$$w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} + \eta\left(\langle x_i x_j\rangle_0 - \langle x_i x_j\rangle_n\right).$$

Here, $p_n(\mathbf{x}; \mathbf{W}) = \mathbf{p}_n = \mathbf{T}^n\mathbf{p}_0$ is the nth-step distribution of the Markov chain with transition matrix[1] $\mathbf{T}$ started at the data distribution $\mathbf{p}_0$.

Restricted Boltzmann machines (Smolensky, 1986; Freund and Haussler, 1992) have connections only between a hidden and a visible unit, i.e., they form a bipartite graph. The energy is then $E(\mathbf{x}, \mathbf{y}; \mathbf{W}) = -\mathbf{y}^T\mathbf{W}\mathbf{x}$, where we have v visible units and h hidden units, and $\mathbf{W} = (w_{ij})$ is an $h \times v$ matrix of real-valued weights. We denote such a machine by RBM(v, h). By making h large, an RBM can be given far more representational power than a VBM, but the log-likelihood can have multiple maxima. The learning is simpler than in a general Boltzmann machine because the visible units are conditionally independent given the hidden units, and the hidden units are conditionally independent given the visible units. One step of Gibbs sampling can therefore be carried out in two half-steps: the first updates all the hidden units and the second updates all the visible units. Equivalently, we can write $\mathbf{T} = \mathbf{T}_\mathbf{x}\mathbf{T}_\mathbf{y}$ where $t_{\mathbf{x};j,i} = p(\mathbf{x} = j|\mathbf{y} = i; \mathbf{W})$ and $t_{\mathbf{y};i,j} = p(\mathbf{y} = i|\mathbf{x} = j; \mathbf{W})$.

Since $\partial E/\partial w_{ij} = -y_i x_j$, ML learning takes the form

$$w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} + \eta\left(\left\langle \langle y_i x_j\rangle_{p(\mathbf{y}|\mathbf{x};\mathbf{W})}\right\rangle_0 - \langle y_i x_j\rangle_\infty\right)$$

while CD_n learning takes the form

$$w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} + \eta\left(\left\langle \langle y_i x_j\rangle_{p(\mathbf{y}|\mathbf{x};\mathbf{W})}\right\rangle_0 - \langle y_i x_j\rangle_n\right).$$

2 Analysis of the fixed points

A probability distribution over v units is a vector of 2^v real-valued components (from 0 to $2^v - 1$ in bi-

[1] Here and elsewhere we omit the dependence of $\mathbf{T}$ on the parameter values $\mathbf{W}$ to simplify the notation.

nary notation) that lives in the 2^v–dimensional simplex $\Delta_{2^v} = \{\mathbf{x} \in \mathbb{R}^{2^v} : \ x_i \geq 0, \ \sum_{i=1}^{2^v} x_i = 1\}$. Each coordinate axis of $\mathbb{R}^{2^v}$ corresponds to a state (binary vector). We write such a distribution as $p(\cdot)$, when emphasising that it is a function, or as a vector $\mathbf{p}$, when emphasising that it is a point in the simplex. We define a Markov chain through its transition operator, which is a stochastic $2^v \times 2^v$ matrix $\mathbf{T}$. We use the Gibbs sampler as the transition operator because of its simplicity and its wide applicability to many distributions. For Boltzmann machines with finite weights, the Gibbs sampler converges to a stationary distribution which is the model distribution $p(\mathbf{x}; \mathbf{W})$. For an initial distribution $\mathbf{p}$ we then have $\mathbf{p}_n = \mathbf{T}^n \mathbf{p}$ for $n = 1, 2, \ldots$; and $p(\mathbf{x}; \mathbf{W}) = \mathbf{p}_\infty = \mathbf{T}^\infty \mathbf{p}$ for any $\mathbf{p} \in \Delta_{2^v}$, i.e., $\mathbf{T}^\infty = \mathbf{p}_\infty \mathbf{1}^T$ where $\mathbf{1}$ is the vector of ones.

VBMs or RBMs define a manifold $\mathcal{M}$ within the simplex, parameterised by $\mathbf{W}$, with $w_{ij} \in \mathbb{R}$ (we ignore the case of infinite weights, corresponding to distributions in the intersection of $\mathcal{M}$ and the simplex boundary). The learning (ML or CD) starts at a point in $\mathcal{M}$ and follows the (approximate) gradient of the log-likelihood, thus tracing a trajectory within $\mathcal{M}$.

For ML with gradient learning, the fixed points are the zero-gradient points (maxima, minima and saddles), which satisfy $\langle \mathbf{g} \rangle_0 = \langle \mathbf{g} \rangle_\infty$ where $\mathbf{g} = \partial E / \partial \mathbf{W}$. For n-step CD, the fixed points satisfy $\langle \mathbf{g} \rangle_0 = \langle \mathbf{g} \rangle_n$. In this section we address the theoretical question of whether the fixed points of ML are fixed points of CD and vice versa. We show that, in general,

$$\exists \mathbf{p}_0 : \ \langle \mathbf{g} \rangle_0 = \langle \mathbf{g} \rangle_\infty \neq \langle \mathbf{g} \rangle_1$$
$$\exists \mathbf{p}_0 : \ \langle \mathbf{g} \rangle_0 = \langle \mathbf{g} \rangle_1 \neq \langle \mathbf{g} \rangle_\infty .$$

We give a brief explanation of a framework for analysing the fixed points of ML and CD; full details appear in Carreira-Perpiñán and Hinton (2004). The idea is to fix a value of the weights and so a value of the moments (defined below), determine which data distributions $\mathbf{p}_0$ have such moments (i.e., the opposite of the learning problem) and then determine under what conditions ML and CD agree over those distributions. Call $\mathbf{G}$ the $|\mathbf{W}| \times 2^v$ matrix of energy derivatives, defined by

$$G_{i\mathbf{x}} = -\frac{\partial E}{\partial w_i}(\mathbf{x}; \mathbf{W})$$

where we consider $\mathbf{W}$ as a column vector with $|\mathbf{W}|$ elements and the state $\mathbf{x}$ takes values $0, 1, \ldots, 2^v - 1$ in the case of v binary variables. We can then write the moments $\mathbf{s} = \left\langle -\frac{\partial E}{\partial \mathbf{W}} \right\rangle_{\mathbf{p}} = -\langle \mathbf{g} \rangle_{\mathbf{p}}$ of a distribution $\mathbf{p}$ as $\mathbf{s} = \mathbf{G}\mathbf{p}$, i.e., $\mathbf{s}$ is a linear function of $\mathbf{p}$. Call $\mathbf{T}$ the transition matrix for the sampling operator with stationary distribution $\mathbf{p}_\infty$ (so we have $\mathbf{p}_\infty = \mathbf{T}\mathbf{p}_\infty$). In general, both $\mathbf{G}$ and $\mathbf{T}$ are functions of $\mathbf{W}$.

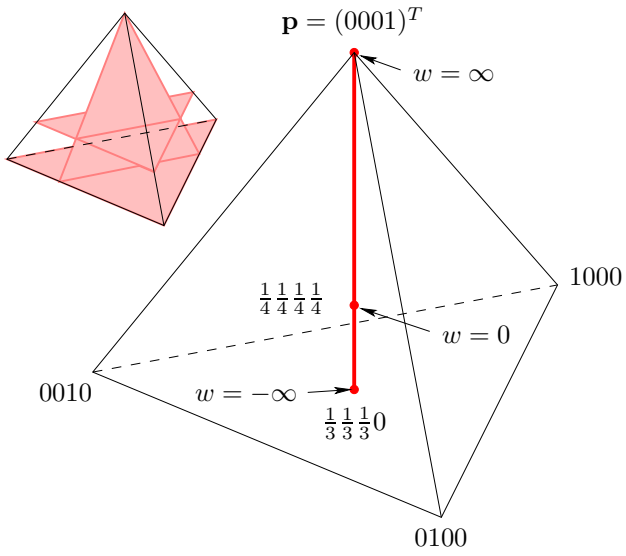


Figure 1: The simplex in 4 dimensions of the VBM(2). The model has a single parameter $\mathbf{W} = w \in (-\infty, \infty)$. The tetrahedron represents the simplex, i.e., the set $\{\mathbf{p} \in \mathbb{R}^4 : \ \mathbf{1}^T \mathbf{p} = 1, \ \mathbf{p} \geq 0\}$. The tetrahedron corners correspond to the pure states, i.e., the distributions that assign all the probability to a single state. The red vertical segment is the manifold of VBMs, i.e., the distributions reachable by a VBM(2) for $w \in (-\infty, \infty)$. The ML estimate of a data distribution is its orthogonal projection on the model manifold. The CD estimate only agrees with the ML one for data distributions in the 3 shaded planes in the inset.

Now consider a fixed value of $\mathbf{W}$ and let $\mathbf{p}_\infty$ be its associated model distribution. Thus its moments are $\mathbf{s}_\infty = \mathbf{G}\mathbf{p}_\infty$. We define two sets $\mathcal{P}_0$ and $\mathcal{P}_1$ that depend on $\mathbf{s}_\infty$. First, the set of data distributions $\mathbf{p}_0$ that have those same moments $\mathbf{s}_\infty$ is:

$$\mathcal{P}_0 = \left\{ \mathbf{p}_0 \in \mathbb{R}^{2^v} : \begin{array}{c} \mathbf{G}\mathbf{p}_0 = \mathbf{s}_\infty \\ \mathbf{1}^T \mathbf{p}_0 = 1 \\ \mathbf{p}_0 \geq 0 \end{array} \right\}.$$

Each distribution in $\mathcal{P}_0$ gives a fixed point of ML. Likewise, define the set of data distributions $\mathbf{p}_0$ whose distribution $\mathbf{p}_1 = \mathbf{T}\mathbf{p}_0$ (first step in the Markov chain, i.e., what CD_1 uses instead of $\mathbf{p}_\infty$) has the same moments as $\mathbf{s}_\infty$:

$$\mathcal{P}_1 = \left\{ \mathbf{p}_0 \in \mathbb{R}^{2^v} : \begin{array}{c} \mathbf{G}\mathbf{T}\mathbf{p}_0 = \mathbf{s}_\infty \\ \mathbf{1}^T \mathbf{p}_0 = 1 \\ \mathbf{p}_0 \geq 0 \end{array} \right\}.$$

Both $\mathcal{P}_0$ and $\mathcal{P}_1$ are nonempty since $\mathbf{p}_\infty$ is in both. Now we can reformulate the problem in terms of the sets $\mathcal{P}_0$ and $\mathcal{P}_1$. For example, a distribution with $\mathbf{p}_0 \in \mathcal{P}_0$ and $\mathbf{p}_0 \notin \mathcal{P}_1$ satisfies $\langle \mathbf{g} \rangle_0 = \langle \mathbf{g} \rangle_\infty \neq \langle \mathbf{g} \rangle_1$ and thus gives a fixed point of ML but not of CD (that is, the CD learning rule would move away from such a $\mathbf{p}_\infty$).

In general (and ignoring technical details regarding the inequality $\mathbf{p}_0 \geq 0$), $\mathcal{P}_0$ and $\mathcal{P}_1$ are linear subspaces of the same dimension because $\mathbf{G}$ is full rank (the moments are l.i.) and $\mathbf{T}$ is generally full rank. Thus we cannot *generally* expect $\mathcal{P}_0 = \mathcal{P}_1$, so points with CD bias are the rule; points in $\mathcal{P}_0 \cap \mathcal{P}_1$ have no CD bias but are the exception (the intersection being a lower-dimensional subspace). We can make the statement precise for a given model. For example, for VBM(2) with Gibbs sampling we have $\mathbf{G} = (0\ 0\ 0\ 1)$ ($\mathbf{G}$ happens to be independent of $\mathbf{W}$ for VBMs), we can compute $\mathbf{T}$ and we can work out the set $\mathcal{P}_0 \cap \mathcal{P}_1$ for every $\mathbf{s}_\infty$ value. The resulting set, which contains all the data distributions for which ML and CD have the same fixed points (i.e., no bias), is the union (intersected with the simplex) of the 3 planes: $p_{11} = 0$; $p_{11} = \frac{1}{4}$; $3p_{01} + p_{11} = 1$, where we write a distribution as a 4-dimensional vector $\mathbf{p} = (p_{00}\ p_{01}\ p_{10}\ p_{11})^T$, corresponding to the probabilities of the 4 states $00, \ldots, 11$ (see fig. 1). This set has measure zero in the simplex, so CD is biased for almost every data distribution.

A reachable distribution $\mathbf{p}_0 \in \mathcal{M}$ is a fixed point for both CD and ML, as it is invariant under $\mathbf{T}$ (Hinton, 2002). This is consistent with the above argument, as $\mathbf{p}_0 = \mathbf{p}_\infty \in \mathcal{P}_0 \cap \mathcal{P}_1$. The distributions of practical interest are typically unreachable because real data are nearly always more complicated than our computationally tractable model of it.

In summary, we expect that for almost every data distribution $\mathbf{p}_0$, the fixed points of ML are not fixed points of CD and vice versa. This means that, in general, CD is a biased learning algorithm. Our argument can be applied to models other than Boltzmann machines, transition operators other than Gibbs sampling, and to $n > 1$ (writing $\mathbf{T}^n$ instead of $\mathbf{T}$). What determines whether CD is biased are the hyperplanes defined by the matrices $\mathbf{G}$ and $\mathbf{GT}$. However, non-trivial models (i.e., defining a lower-dimensional manifold) may exist for which CD is not biased; an example is Gaussian Boltzmann machines (Williams and Agakov, 2002) and Gaussian distributions, at least in 2D (Carreira-Perpiñán and Hinton, 2004).

This analysis does not imply that CD learning converges (to a stable fixed point); at present, we do not have a proof for this. But if CD does converge, as it appears to in practice and in all our experiments, it can only converge to a fixed point. Naturally, ML does converge to its stable fixed points (maxima) from almost everywhere, since it follows the exact gradient of an objective function; in the noisy sampling case that is used in practice, it also converges provided the learning rate η follows a Robbins-Monro schedule (Benveniste et al., 1990), since the rule performs stochastic gradient learning.

3 Experiments with fully visible BMs

Since CD is biased with respect to ML for almost all data distributions, we now investigate empirically the magnitude of the bias. In all experiments in the paper, ML and CD are tested under exactly the same conditions (unless otherwise stated). Both ML and CD learning use the same initial weight vectors, the same constant learning rate $\eta = 0.9$ and the same maximum of 10 000 iterations (which is rarely reached for VBMs), stopping when $\|\mathbf{e}\|_\infty < 10^{-7}$ (where $\mathbf{e} = \langle x_i x_j \rangle_0 - \langle x_i x_j \rangle_\infty$ is the gradient vector for ML, and $\mathbf{e} = \langle x_i x_j \rangle_0 - \langle x_i x_j \rangle_1$ is the approximate gradient for CD_1). All the experiments use $n = 1$ step of Gibbs sampling with fixed ordering of the variables for CD learning, because this should produce the greatest bias (since CD $\xrightarrow[n\to\infty]{}$ ML). Although each of our simulated models is necessarily small, our empirical results hold for a range of model sizes and conditions, which suggests they may be more generally valid.

In this section we consider fully visible Boltzmann machines, denoted VBM(v), which have a single ML optimum. It appears that CD has a single convergence point too: for $v = 2$ we can prove this, and for $v \in \{3, \ldots, 10\}$ we checked empirically by running CD from many different initial weight vectors that it always converged to the same point (up to a small numerical error). Thus, we assume that CD has a unique convergence point for VBM(v). This allows us to characterise the bias for this model class by sampling many data distributions and computing the convergence point of ML and of CD.

For a given value of v we sampled a number (as large as computationally feasible) of data distributions uniformly distributed in the simplex in 2^v variables (see Carreira-Perpiñán and Hinton, 2004 for details of how to generate these samples). Then we ran ML and CD starting with $\mathbf{W} = \mathbf{0}$ because small weights give faster convergence on average. The results for experiments for $v \in \{2, \ldots, 10\}$ were qualitatively similar. For $v = 2$ it was feasible to sample 10^4 data distributions and the results are summarised in figures 2–5.

The histograms in figs. 2–3 show the bias is very small for most distributions. Fig. 4 shows that the KL error (for both ML and CD) is small for data distributions near the simplex centre. For less vague data distributions there is more variability, with some distributions having a low error and some having a much higher one. Generally speaking, the distributions having the highest KL error for ML (i.e., the distributions that are modelled worst by the VBM) are also the ones that have the highest bias. Most of these lie near the boundaries of the simplex, particularly near the corners. However, not all corners and boundaries are far

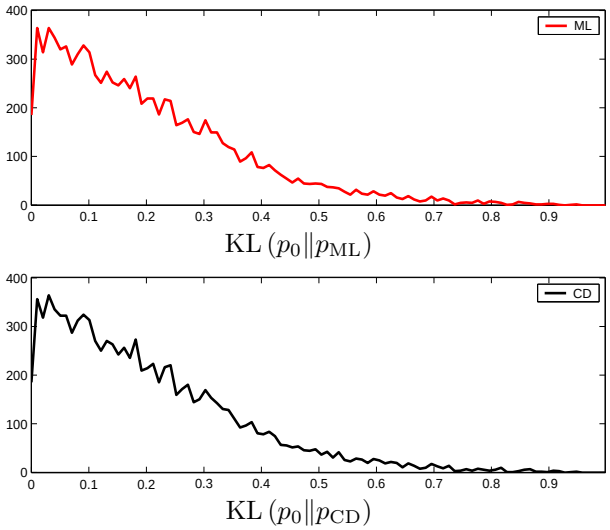


Figure 2: Histograms of $\mathrm{KL}(p_0 \| p_{\mathrm{ML}})$ and $\mathrm{KL}(p_0 \| p_{\mathrm{CD}})$ for VBM(2) after learning on 10^4 data distributions, where p_{ML} and p_{CD} are the convergence points for ML and CD, respectively. The performance of CD is very close to ML on average.

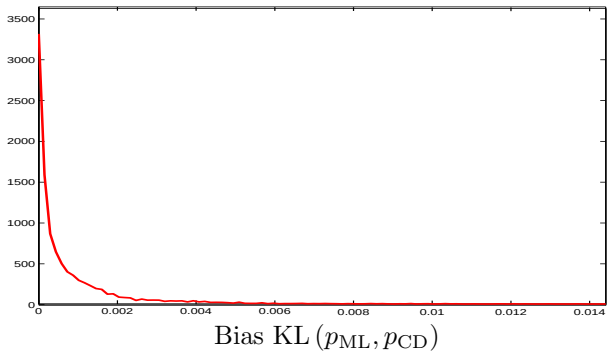


Figure 3: Histogram of the symmetrised KL divergence for VBM(2) between the model distributions found by ML and CD for all 10^4 data distributions. This shows that the bias of CD is almost always very small ($< 5\%$ of the KL error obtained by ML for the same distribution; data not shown). However, data distributions do exist that have a relatively large bias.

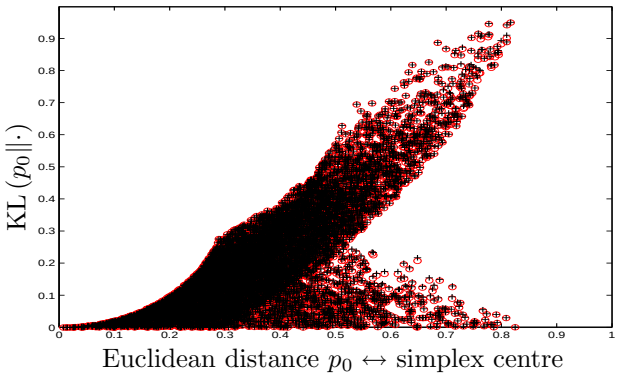


Figure 4: KL error for ML $\mathrm{KL}(p_0 \| p_{\mathrm{ML}})$ (red $\circ$) and for CD $\mathrm{KL}(p_0 \| p_{\mathrm{CD}})$ (black $+$) vs Euclidean distance $\|p_0 - u\|$ between the data distribution and the uniform distribution (centre of the simplex). This Euclidean distance gives a linear ordering of the data distributions (lowest Euclidean distance: p_0 is the uniform distribution, $\|p_0 - u\| = 0$; highest: p_0 is one of the corners of the simplex, $\|p_0 - u\| = \sqrt{1 - 2^{-v}}$). For clarity, not all 10^4 distributions are plotted.

from the model manifold; this depends on the geometry of the model. In fig. 4 (for $v = 2$) we can discern the geometry of the simplex in fig. 1. The discontinuity in the slope at a Euclidean distance $\|p_0 - u\|$ just less than 0.3 corresponds to the radius of the inscribed sphere. The branch in the scatterplot which has low error corresponds to the direction passing through the centre and the simplex corner corresponding to the delta distribution of the $(1, 1)$ state (i.e., along the VBM manifold). The other branch which has high error and more data points corresponds to the directions passing through the centre and any of the other 3 corners (i.e., away from the VBM manifold).

As v increases, most of the volume of the simplex concentrates at a distance intermediate between the corners and the centre, close to the radius of the inscribed hypersphere. Consequently, a finite uniform sample contains essentially no points near the boundaries of the simplex, which produce the highest bias. For large v, CD has very small bias for nearly all randomly chosen data distributions. Only those rare distributions near the simplex boundaries produce a significant bias, but these are important in practice: real-world distributions are often near the boundaries (though not as far as the corners) because large parts of the data space have negligible probability.

Fig. 5 shows some typical learning curves. Both CD and ML decrease in a similar way, converging at the same rate (first-order), taking the same number of iterations to converge to a given tolerance. CD yields a higher KL error. In the lower example, the CD curve increases slightly at the end, suggesting it came close to the ML optimum but then moved away.

In summary, we find the CD bias to be very small for most distributions and to be highest (but still small) with real-world distributions (near the simplex boundary). This bias is small in relative terms (compared to the KL error for ML) and absolute terms (compared to the simplex dimensions). CD and ML converge at about the same rate, but an ML iteration costs much

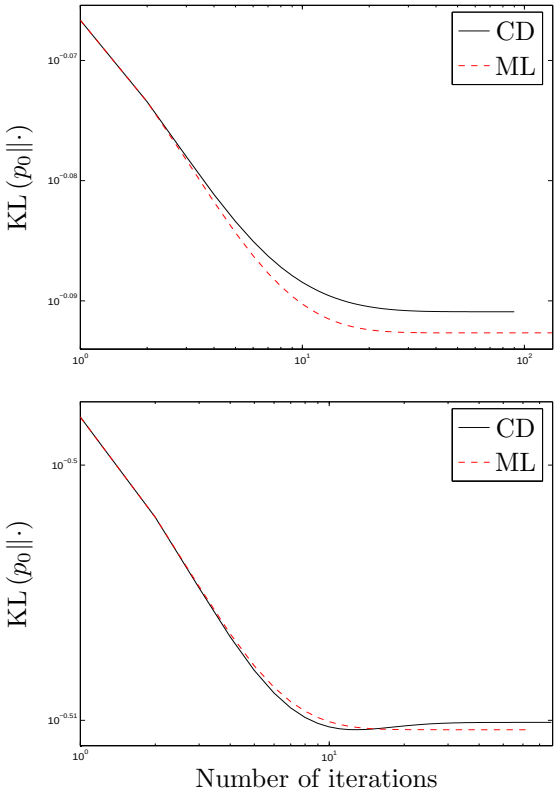


Figure 5: Learning curves for ML and CD for 2 randomly chosen data distributions. Axes are in log scale.

more than a CD one in a MCMC implementation.

4 Experiments with restricted BMs

RBMs are practically more interesting than VBMs, since they have a higher representational power. They also introduce a new element that complicates our study: the existence of multiple local optima of ML and CD. This prevents the characterisation of the bias over a large number of data distributions. Instead, we can only afford to select one data distribution (or a few) and try to characterise the set of all optima of ML and CD.

Given a data distribution, we generate a collection of 60 random initial weight vectors $\mathbf{W}$ and compute all the optima of ML and CD that are reachable from any of the initial weight vectors or from the optima found by the other learning method. This requires iterating over the current set of optima with ML and CD, until no new optima are found. The result is a bipartite, self-consistent convergence graph where an arrow $A \to B$ indicates that ML optimum A converges to CD optimum B under CD, or CD optimum A converges to ML optimum B under ML. Using many different initial weight vectors should give a representative collection of optima and a Good-Turing estimator (Good, 1953) can be used as a coarse indicator of how many optima we missed. The graph depends on how we decide whether two very similar optima are really the same. The threshold and number of parameter updates have to be carefully chosen so that truly different optima are not confused but two discoveries of the same optimum are not considered different. We found that using the symmetrised KL distance with a threshold of 0.01 worked well with 10^5 parameter updates.

We ran experiments for various values of v and h and various data distributions. Fig. 6 summarises the results for one representative case, corresponding to: $v = 6$, $h = 4$. The data distribution was generated from a data set of 4 binary vectors by adding an extra count of 0.1 to each possible binary vector and renormalising (thus it is close to the simplex boundary). We used 10^5 iterations and 60 different initial weight vectors: 20 random $\mathcal{N}(0, \sigma = 0.1)$, 20 random $\mathcal{N}(0, \sigma = 1)$ and 20 random $\mathcal{N}(0, \sigma = 10)$. We found 27 ML optima and 28 CD optima, and missed about 3 and 4, respectively (Good-Turing estimate).

Panel **A** shows a 2D visualisation of ML optima (red $\circ$) and CD optima (black $+$), i.e., the visible-unit distributions p_{ML} and p_{CD}, and their convergence relations. The blue $\star$ is the data distribution p_0 (of the visible variables). To avoid cluttering the plot, pairs of arrows $A \leftrightarrows B$ are drawn as a single line without arrowheads $A - B$. Note that many such lines are too short to be distinguished. The 2D view was obtained with SNE (Hinton and Roweis, 2003) which tries to preserve the local distances. Using a perplexity of 3 to determine the local neighborhood size, SNE gives a better visualisation than projecting onto the first 2 principal components.

This panel shows an important and robust phenomenon: ML and CD optima typically come in pairs that converge to each other. The CD optimum always has a greater or equal KL error than its associated ML optimum but the difference is small. These pairs are to be expected for CD_n when n is large because CD becomes ML as $n \to \infty$. However they occur very often even for $n = 1$, as shown. Panels **B**–**C** show that the choice of initial weights has a much larger effect on the KL error than the CD bias.

5 Using CD to initialise ML

The previous experiments show that CD takes us close to an ML optimum, but that a small bias remains. An obvious way to eliminate this bias is to use increasing values of n as training progresses. In this section we explore a crude version of this strategy: run CD until it is close to convergence then use a short run of ML

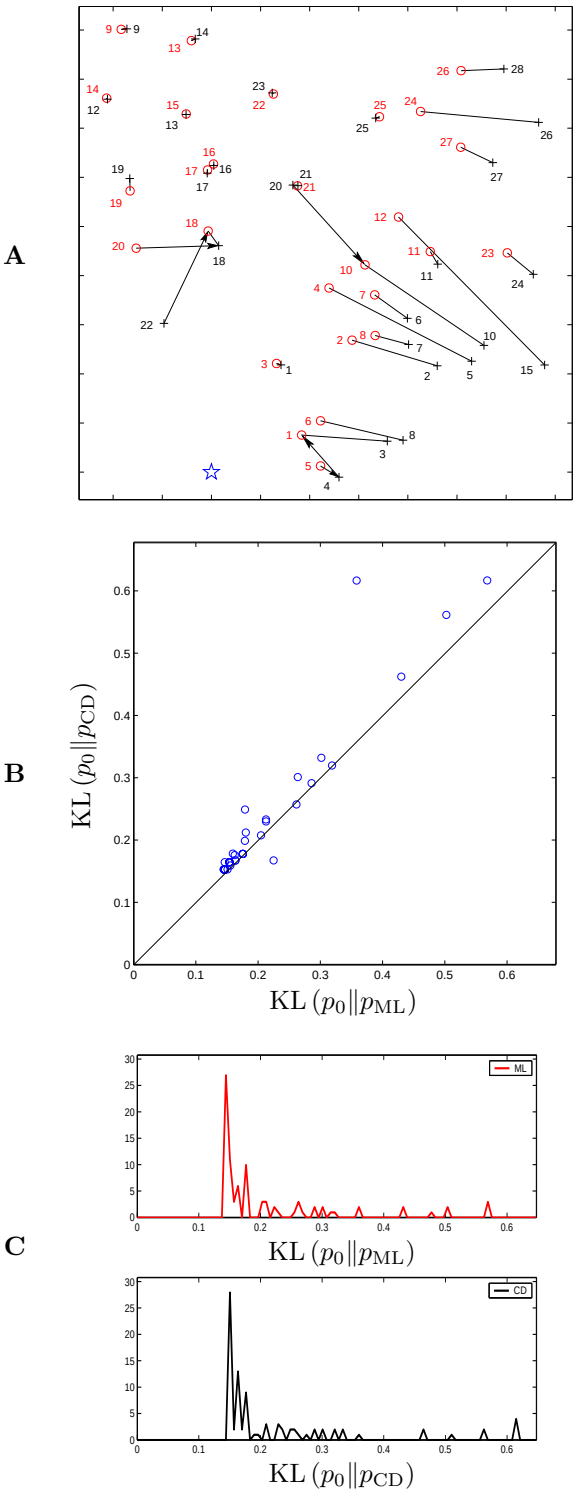


Figure 6: Empirical study of the convergence points of ML and CD for RBM$(6, 4)$ with a single data distribution. **A**: 2D SNE visualization of the points (ML: red ○; CD: black +), and convergence relations among them with ML and CD (a line without an arrowhead stands for two arrows ⇆, to avoid clutter). **B**: KL error of CD vs ML from the same initial weight vectors, for 60 random initial weight vectors. **C**: histograms of the KL error of ML and CD.

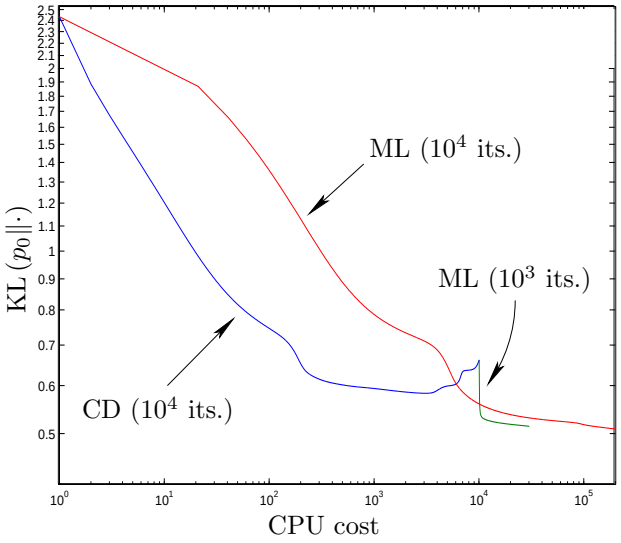


Figure 7: Learning curves for CD-ML and ML, where each ML iteration is scaled to cost as much as 20 CD iterations. The axes have a log scale, so CD-ML is an order of magnitude faster for about the same final KL.

to reduce the bias. We call this strategy CD-ML. We use a data distribution which is more representative of a real problem. It is located on the simplex boundary and derived from the statistics of all 3×3 patches in 11000 16×16 images of handwritten digits from the USPS dataset. The 256 intensity levels are thresholded at 150 to produce 9-dimensional binary vectors (thus $v = 9$). p_0 is the normalised counts of each of these binary vectors in the 2 156 000 patches.

We used 60 different initial weight vectors: 20 random $\mathcal{N}(0, \sigma = \frac{1}{3})$, 20 random $\mathcal{N}(0, \sigma = 1)$ and 20 random $\mathcal{N}(0, \sigma = 3)$. For each starting condition, two types of learning were used: ML learning for 10^4 iterations; and CD learning for 10^4 iterations, followed by a shorter run of ML learning. We ran experiments for $h \in \{1, \ldots, 8\}$ and found that there is a unique ML optimum and several CD optima of varying degrees of bias. If CD learning was followed by 1 000 iterations of ML, all the CD optima converged to the ML optimum.

Fig. 7 shows the learning curves (i.e., the error KL $(p_0 \| \cdot)$ as a function of estimated CPU time) for the different methods with $h = 8$: CD (blue line), the short ML run (1 000 iterations) following CD (green line) and ML (red line), for a selected starting condition. We assume that each ML iteration costs 20 times as much as each CD iteration (a reasonable estimate for the size of this RBM). CD-ML reaches the same error as ML but at a small fraction of the cost. Note how sharply the CD-ML curve drops when we switch to ML, suggesting good performance can be achieved with very few of the expensive ML iterations.

6 Conclusion

Our first result is negative: for two types of Boltzmann machine we have shown that, in general, the fixed points of CD differ from those of ML, and thus CD is a biased algorithm. This might suggest that CD is not a competitive method for ML estimation of random fields. Our remaining, empirical results show otherwise: the bias is generally very small, at least for Gibbs sampling, since CD typically converges very near an ML optimum. And this small bias can be eliminated by running ML for a few iterations after CD, i.e., using CD as an initialisation strategy for ML, with a total computation time that is much smaller than that of full-fledged ML (which will also have slight bias because the Markov chain cannot be run forever).

The theoretical analysis of CD is difficult because of the complicated form that the p_1 (or p_n) distribution takes; p_1 is a moving target that changes with $\mathbf{W}$ in a complicated way, and depends on the sampling scheme used (e.g. Gibbs sampling). As a result, very few theoretical results about CD exist. MacKay (2001) gave some examples of CD bias, but these used unusual sampling operators. Our analysis applies to any model and operator (through the $\mathbf{G}$ and $\mathbf{T}$ matrices), in particular generally applicable operators such as Gibbs sampling. Williams and Agakov (2002) showed that, for 2D Gaussian Boltzmann machines, CD is unbiased and typically decreases the variance of the estimates. Yuille (2004) gives a condition for CD to be unbiased, though this condition is difficult to apply in practice.

One open theoretical problem is whether the exact version of CD converges (we believe that it does). Assuming we can prove convergence for the exact case, the right tools to use to prove it in the noisy case are probably those of stochastic approximation (Benveniste et al., 1990; Yuille, 2004).

Acknowledgements

This research was funded by NSERC and CFI. GEH is a fellow of CIAR and holds a CRC chair.

References

A. Benveniste, M. Métivier, and P. Priouret. *Adaptive Algorithms and Stochastic Approximations*. Springer-Verlag, 1990.

M. Á. Carreira-Perpiñán and G. E. Hinton. On contrastive divergence (CD) learning. Technical report, Dept. of Computer Science, University of Toronto, 2004. In preparation.

H. Chen and A. F. Murray. Continuous restricted Boltzmann machine with an implementable training algorithm. *IEE Proceedings: Vision, Image and Signal Processing*, 150(3):153–158, June 20 2003.

Y. Freund and D. Haussler. Unsupervised learning of distributions on binary vectors using 2-layer networks. In *NIPS*, pages 912–919, 1992.

W. Gilks, S. Richardson, and D. J. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1996.

I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3/4):237–264, Dec. 1953.

X. He, R. S. Zemel, and M. Á. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *CVPR*, pages 695–702, 2004.

G. Hinton and S. T. Roweis. Stochastic neighbor embedding. In *NIPS*, pages 857–864, 2003.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, Aug. 2002.

S. Z. Li. *Markov Random Field Modeling in Image Analysis*. Springer-Verlag, 2001.

D. J. C. MacKay. Failures of the one-step learning algorithm. Available online at `http://www.inference.phy.cam.ac.uk/mackay/abstracts/gbm.html`, 2001.

R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG–TR–93–1, Dept. of Computer Science, University of Toronto, Sept. 1993. Available online at `ftp://ftp.cs.toronto.edu/pub/radford/review.ps.Z`.

P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart and J. L. MacClelland, editors, *Parallel Distributed Computing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*, chapter 6. MIT Press, 1986.

Y. W. Teh, M. Welling, S. Osindero, and G. E. Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4:1235–1260, Dec. 2003.

C. K. I. Williams and F. V. Agakov. An analysis of contrastive divergence learning in Gaussian Boltzmann machines. Technical Report EDI–INF–RR–0120, Division of Informatics, University of Edinburgh, May 2002.

G. Winkler. *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods*. Springer-Verlag, second edition, 2002.

A. Yuille. The convergence of contrastive divergences. To appear in NIPS, 2004.

OOBN FOR FORENSIC IDENTIFICATION THROUGH SEARCHING A DNA PROFILES' DATABASE

David Cavallini
Department of Statistics "G. Parenti"
University of Florence (Italy)
cavallin@ds.unifi.it

Fabio Corradi
Department of Statistics "G. Parenti"
University of Florence (Italy)
corradi@ds.unifi.it

Abstract

In this paper we evaluate forensic identification hypotheses conditionally to the characteristics observed both on a crime sample and on individuals contained in a database. First we solve the problem via a computational efficient Bayesian Network obtained by transforming some recognized conditional specific independencies into conditional independencies. Then we propose an Object Oriented Bayesian Network representation, first considering a generic characteristic, then inheritable DNA traits. In this respect we show how to use the Object Oriented Bayesian Network to evaluate hypotheses concerning the possibility that some unobserved individuals, genetically related to the individuals profiled in the database, are the donors of the crime sample.

1 INTRODUCTION

Bayesian Networks (BN) are a powerful and compact representation of complex statistical models that exploit some recognized conditional independencies among random variables. A BN is defined as a pair of objects: a Directed Acyclic Graph (DAG) whose nodes represent discrete random variables, and a set of Conditional Probability Tables (CPT) which defines the conditional distributions of each vertex given the parents.

One of the reasons to represent a statistical model as a BN is the possibility to use well-established and effective algorithms to solve the inferential issue, i.e. to compute the distribution of some variables of interest conditionally to the evidence by using one of the available propagation algorithms (e.g. Jensen, 2001).

A limit in the representation of a BN arises when the number of random variables in the model increases due to some features of the problem.

Typically, this happens for time series models where a certain structure, a time-slice, is replicated over time, so that links between random variables in different time slices are established. This also occurs when we are interested in the relations between sets of random variables and when some specified relations between the sets must be taken into account. In the former case the model increases its dimensions over time, in the latter its growth depends on the number of sets involved.

In this respect, a new approach, stemmed from the *Object Oriented* language, has been introduced in the last few years. This modelling tool, called *Object Oriented Bayesian Network*, provides a useful technique capable of building a BN by merging pieces of simple BNs. Each item is an instantiation of a well-defined class which can be modified in order to accomplish the maintenance requirements. An update in the structure or in the CPTs of a class is automatically extended to all instantiations of that class. The subject is developed in Koller and Pfeffer (1997) and Bangso and Wuillemin (2000).

Here, we specifically deal with the forensic identification problem arising when a crime sample has been found but there is no clue about its origin. Searching a database (DB) of previously collected items is a common practice and the scope of this analysis is to assess the probability for each member of the database to be the origin of the trace. The problem has found considerable attention in the literature, but only not inheritable characteristics were considered, see e.g. Stockmarr (1999), Donnelly and Friedman (1999), Dawid (2001) and Meester and Sjerps (2003).

The aim of this paper is to show how this dimension-dependent problem, once opportunely formulated as a BN, can be effectively tackled. First, we provide a theoretical contribution transforming some recognized conditional specific independencies (Geiger and Heckerman, 1996) into conditional independencies, Section (2). Then, since the resulting BN shows many different repetitive structures, we propose an OOBN solution. The use of OOBN to model genetic data

for identification was previously experienced by Dawid (2003) with special attention to the possibility of mutations.

The DB search problem is first developed for a not inheritable characteristic, but our real aim is to consider more complex genetic traits in order to extend the search to the relatives of the individuals profiled in the database, providing hints also when no match between the crime sample and one (or more) of the database members is found, Section (3.2). In Section (4) we provide the results of a simulation study based on a real database, emphasizing some computational issues. Finally we draw some conclusions.

2 EQUIVALENT BN FOR THE DB SEARCH PROBLEM

Let X the discrete population characteristic (or attribute) considered for the forensic identification problem. With $\mathcal{X}$ we indicate the set of the m states of X. The parameter θ_x, with $x \in \mathcal{X}$, is the probability that X is in state x, that is $P(X = x) = \theta_x$ and $\sum_{x\in\mathcal{X}} \theta_x = 1$. Uncertainty about these probabilities, derived from an inference process, could be introduced but this will not be considered here.

Let N the size of the reference population and n the number of the individuals in the DB. For each of them we define a random variable X_j with $j \in \mathcal{J} = \{1, 2, \ldots, n\}$. Also, we define X_c, the characteristic related to the crime scene, and the hypothesis variable H which has $n + 1$ states. The first n of them represent the originator status of each individual, i.e. $H = j$, with $j \in \mathcal{J}$, means that the origin of the trace is the j-th individual in the DB, while the last, $H = \mathtt{r}$, is referred to the hypothesis that the trace's donor is outside the DB.

To specify the DB search model we adopt some common and reasonable assumptions:

i. the individual characteristics in the DB are jointly independent;

ii. the individual characteristics are jointly independent of the hypothesis variable, i.e. $\mathbf{X} \perp\!\!\!\perp H$ where $\mathbf{X} = \{X_j : j \in \mathcal{J}\}$;

iii. if the individual j is the originator of the trace the crime sample is observed without error $X_j \equiv X_c \mid H = j$;

iv. for $H = \mathtt{r}$ the individual attributes are jointly independent of the characteristic involved in the crime scene, i.e., $\mathbf{X} \perp\!\!\!\perp X_c \mid H = \mathtt{r}$ and $P(X_c = x \mid H = \mathtt{r}) = \theta_x$ with $x \in \mathcal{X}$;

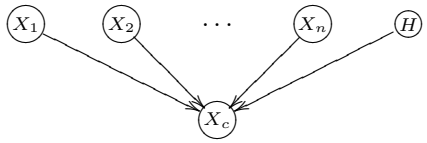


Figure 1: A DAG for the DB search problem.

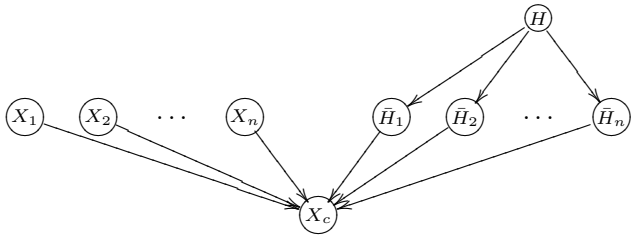


Figure 2: The augmented DAG obtained from Figure (1).

v. no other clue is available in advance, so the prior probability on H is $P(H = j) = 1/N$ and $P(H = \mathtt{r}) = 1 - n/N$.

The graphical structure, depicted in Figure (1), derives only from the assumptions (i) and (ii) while the CPTs are specified according to the assumptions (iii)-(v). Note that (iii) and (iv) imply a whole set of $n + 1$ independence statements: for each value of H a different assertion of independence holds. This form of independence is known as *Conditional Specific Independence* (CSI) (Geinger and Heckerman, 1996), which differs from the usual definition of conditional independence since, in the latter, the independence assertions between variables do not vary according to the values of the conditioning sets.

The proposed network does not feature any conditional independence, so, for some evidence, the probability updating does not take advantage of the graphical representation. Moreover, the size of the CPT of X_c increases exponentially according to the number of individuals in the DB, so that the propagation becomes rapidly unfeasible. Our scope is to provide a more efficient solution by introducing a set of instrumental nodes in order to allow local computations. The result is attained in three steps.

Step 1. First, a set of binary random variables $\bar{\mathbf{H}} = \{\bar{H}_j : j \in \mathcal{J}\}$ is added and a new network is defined on the augmented domain, as in Figure (2).

The marginal distribution of the variables X_j and H does not change with respect to the original network

and the remaining CPTs are defined as follows:

$$\hat{P}(\bar{H}_j = 1 \mid H = i) = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$\hat{P}(X_c \mid \mathbf{X}, \bar{\mathbf{H}} = \bar{\mathbf{h}}) = \begin{cases} P(X_c \mid X_j, H = j) & \text{if } \bar{\mathbf{h}} = \mathbf{1}_j \\ P(X_c \mid H = \mathtt{r}) & \text{if } \bar{\mathbf{h}} = \mathbf{0} \end{cases} \quad (2)$$

where $\mathbf{0}$ and $\mathbf{1}_j$ are vectors of size n. Each element of $\mathbf{0}$ is 0 while the i-th element of $\mathbf{1}_j$ is 0 $\forall i \neq j$ and 1 for $i = j$.

The CPTs for each node $\bar{H}_j$, specified as in (1), are the probabilistic translation of the deterministic logical `if-then` relation, i.e., $\forall j$ `if` $H = j$ `then` $\bar{H}_j = 1$ and $\forall i \neq j$, $\bar{H}_i = 0$. Thus, each variable $\bar{H}_j$ represents the originator status for the j-th individual and the deterministic relation is a consequence of the assumption that the characteristic observed on the crime scene was left by only one individual belonging to the reference population.

It is easy to prove that:

$$\sum_{\bar{\mathbf{H}}} \hat{P}(X_c, \mathbf{X}, \bar{\mathbf{H}}, H) = \sum_{j=1}^{n} \hat{P}(X_c, \mathbf{X}, \bar{\mathbf{H}} = \mathbf{1}_j, H) + \hat{P}(X_c, \mathbf{X}, \bar{\mathbf{H}} = \mathbf{0}, H) = P(X_c, \mathbf{X}, H). \quad (3)$$

Since the hypotheses are mutually exclusive, all configurations of $\bar{\mathbf{H}}$ not equal to the $\mathbf{1}_j$s and $\mathbf{0}$ have zero probability to realize. For this reason, in the marginalization (3), we consider only the relevant configurations of $\bar{\mathbf{H}}$.

The main consequence of the above mentioned result concerns the updating of the query variable H. In fact, for any evidence on $\mathbf{X}$ and X_c, the posterior probability of the hypotheses variable can be calculated indifferently by using the BNs of Figure (1) or Figure (2).

Step 2. Here a *divorcing* technique (Jensen, 2001) is applied. The idea is to introduce a set of mediating variables between parents and children in a large converging connection to lead some parents to divorce. The main advantage of this method is the reduction of the computational efforts because the original clique, $\{\mathbf{X}, X_c, \mathbf{H}\}$, is broken into a tree of smaller cliques.

A reasonable way to divorce the parents of node X_c in Figure (2)'s network is to add n mediating variables $\mathbf{Z} = \{Z_j : j \in \mathcal{J}\}$, which take values in $\mathcal{X}$, so that each pair of variables X_j and H_j are married. Figure (3) illustrates the DAG after divorcing. There, the node $X_c^\star$ represents the characteristic related to the crime

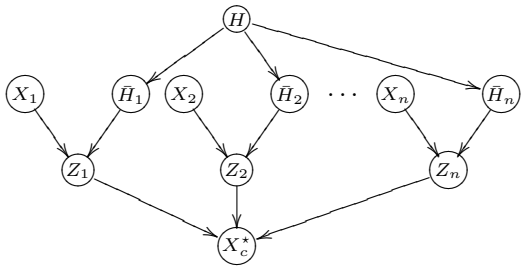


Figure 3: The augmented DAG of Figure (2) after the divorce.

scene which has been redefined for convenience. In particular $X_c^\star$ takes values in $\mathcal{X}^\star = \mathcal{X} \cup \{\text{NA}\}$ where the state labelled NA is an instrumental event to make the conditional distribution of $X_c^\star$ well defined also for $\mathbf{Z}$ values different from those allowed in this context. The $\mathbf{Z}$ can be considered as *private* copies of the crime sample, reproducing its value for each of the members of the DB.

The CPTs specification of the nodes $\mathbf{X}$, $\bar{\mathbf{H}}$ and H remains unchanged with respect to the BN of Figure (2). Imposing the CSI conditions

$$\forall j, Z_j \perp\!\!\!\perp X_j \mid \bar{H}_j = 0, \quad (4)$$

the rest of CPTs are specified as follows

$$\tilde{P}(Z_j = x \mid \bar{H}_j = 0) = \theta_x \quad (5)$$

$$\tilde{P}(Z_j = x \mid X_j = \hat{x}, \bar{H}_j = 1) = \begin{cases} 1 & \text{if } x = \hat{x} \\ 0 & \text{if } x \neq \hat{x} \end{cases} \quad (6)$$

$$\tilde{P}(X_c^\star = \bar{x} \mid \mathbf{Z} = \mathbf{z}) = \begin{cases} 1 & \text{if } \bar{x} = \text{NA or } \forall j, \bar{x} = z_j \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

where $\bar{x} \in \mathcal{X}^\star$ and $x, \hat{x}, z_j \in \mathcal{X}$.

The following proposition provides the probabilistic relation between the networks in Figure (2) and Figure (3).

PROPOSITION 2.1 *For each $x \in \mathcal{X}$ and for a given quantity $C(x)$, depending on x, the following relation holds:*

$$\hat{P}(X_c = x, \mathbf{X}, \bar{\mathbf{H}}, H) = C(x) \cdot \sum_{\mathbf{Z}} \tilde{P}(X_c^\star = x, \mathbf{X}, \bar{\mathbf{H}}, H, \mathbf{Z}) \quad (8)$$

Proof in the Appendix.

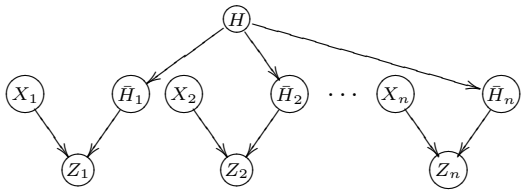


Figure 4: The network obtained by dropping the $X_c^\star$ node and the related incidental arcs from the DAG in Figure (3)

Finally, combining (3) with (8), we obtain the main result:

$$P(X_c = x, \mathbf{X}, H) = C(x) \cdot \sum_{\mathbf{Z},\bar{\mathbf{H}}} \tilde{P}(X_c^\star = x, \mathbf{X}, \bar{\mathbf{H}}, H, \mathbf{Z}) \quad (9)$$

The above equation establishes that for calculating the posterior probability of the hypotheses variable H we can use the network of Figure (4) instead of that in Figure (3).

Step 3. As explained in the proof of **PROPOSITION** 2.1, during the propagation each valid evidence on $X_c^\star$ is transferred to all mediating variables. So, operationally, we build a new DAG merely by dropping the node $X_c^\star$ as well as its incidental arcs, Figure (4). Moreover, we use the characteristic observed on the crime scene for evidencing each vertex Z_j.

3 OOBN FOR THE DB SEARCH

The graph depicted in Figure (4) is conspicuous for its repetitive structure. For each individual profile in the DB the same BN is built and all the networks are mixed by the hypotheses variable H which is the only parent of every $\bar{H}_j$. Therefore, a set of conditional independence assertions appears, i.e., given H, each triple (Z_j, $\bar{H}_j$, X_j) is independent of the rest of the variables so that, for calculating the posterior distributions of H, local computations are allowed.

3.1 NOT INHERITABLE TRAITS

A more compact representation can be achieved by transforming the proposed network into the OOBN framework. As in Bangso and Wuillemin (2000), we define a class, $\mathbb{F}$, containing a simple BN, $\bar{H} \rightarrow Z \leftarrow X$, where the node $\bar{H}$ is an input node while X and Z are interior nodes. For each instantiation of the class $\mathbb{F}(j)$, with $j \in \mathcal{J}$, we build a binary random variable $\bar{H}_j^r$ which is *referenced* node of $\mathbb{F}(j).\bar{H}$. They are connected through a *reference* link ($\Rightarrow$), that is $\bar{H}_j^r \Rightarrow \mathbb{F}(j).\bar{H}$. Moreover, a set of arcs from the general hypotheses variable H pointing towards each referenced node is drawn. Finally, the CPTs related to the variables $\bar{H}_j^r$ are specified as in (1).

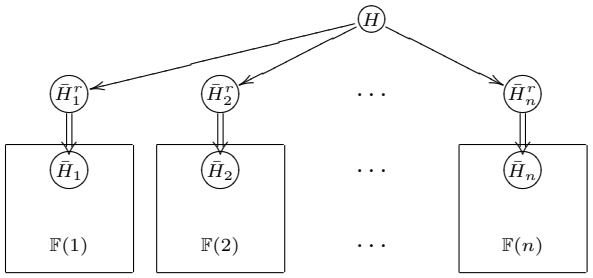


Figure 5: The OOBN representation for the DB search problem derived from Figure (4).

Figure (5) illustrates the OOBN representation for the DB search problem as the basic model to solve the forensic identification issue.

3.2 INHERITABLE DNA TRAITS

A DNA profile involves measurements on several well-specified locations of the DNA, called *loci*. For each locus we observe a genotype i.e. two alleles, one inherited from the father and the other from the mother, even if their origin is not distinguishable. For a generic locus we define two random variables A_0 and A_1 whose states, $a_1, a_2, \ldots, a_m$, are the inherited alleles. In addition, we consider a further random variable X whose states represent the genotypes, i.e., an ordered pair of alleles (a_t, a_u) with $t \leq u$. In this paper we assume Hardy-Weinberg (H-W) conditions and linkage equilibrium. H-W implies that parents are not related so that the inherited alleles in a genotype are independent. Linkage equilibrium refers to the independence among loci in the same individual. This is justified since the loci considered for identification are chosen far enough in the genome to make plausible that they are generated by different meiosis processes.

The genetic inheritance allows us to consider, as the possible donors of the crime sample, also individuals never typed but related to the DB members. In this way the no-match case, the most common in practice, but unfortunately the less useful, could originate *compatible* unobserved individuals, i.e. those having a positive probability for the characteristic observed on the crime sample, conditional to all the available evidence. For instance, a DB member not matching the crime sample but sharing at least one allele for each considered locus has a compatible child.

Here, we consider a pedigree, $\mathcal{F}$, constituted by a generic individual (i), their parents (0 and 1), their child (c), their partner (p) and their brother (b). Note

that Labels 0 and 1 refer to a generic parent and not specifically to the mother or father because this information is not available. Since each pedigree is built around a member of the DB we call it a *first-degree-relative* pedigree. This choice is essentially due to the fact that, in the expectation of a significant hint about the trace's donor, we cannot explore too far from each individual in the DB. Refinements of the search could be achieved if familiar connections between the DB members were known. This kind of information is not usually recorded in a DB but, if available, could be exploited to relate two or more familiar classes with suitable links.

In this new perspective, the variables H and H_j^r shown in Figure (5) have a new meaning.

The j-th state of H, with $j \in \mathbb{J}$, refers to the hypothesis that the donor of the trace belongs to the family of the j-th individual of the DB, while $H = \mathtt{r}$ concerns the possibility that the trace was left by someone not included in the considered families. Every variable $\bar{H}_j^r$ takes values in $\bar{\mathcal{F}} = \mathcal{F} \cup \mathtt{r}$. The state $\mathtt{r}$ concerns the hypothesis that the trace is left by none of the considered family's members, while the statement $\bar{H}_j^r = q$, with $q \in \mathcal{F}$ means that the donor of the trace is the q-th member of the jth family.

Since, by definition, we have no clue about the donor of the trace, all the considered individuals are assumed to have the same prior probability to be the searched person. Within each family we assume that six persons are the possible suspects but, obviously, some of them might be ruled out if, e.g., they were in jail or dead. To refine the analysis we define an indicator variable $\mathrm{J}_{h,j} \in \{0,1\}$ for the relevance of the q-th person in the j-th family. Moreover, with $k_j = \sum_{q \in \mathcal{F}} \mathrm{J}_{q,j}$ we indicate the number of the relevant persons in the j-th family. The prior on H is $P(H = j) = k_j/N$, and

$$P(\bar{H}_j^r = q \mid H = i) = \begin{cases} \mathrm{J}_{q,j}/k_j & \text{if } j = i \text{ and } q \neq \mathtt{r} \\ 1 & \text{if } j \neq i \text{ and } q = \mathtt{r} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $i, j \in \mathbb{J}$ and $q \in \mathcal{F}$.

For inheritable DNA traits the class $\mathbb{F}$ includes the first-degree-relative pedigree and the set of hypotheses variables related to a generic family. Considering the *Allele Network* proposed by Lauritzen and Sheehan (2003), we provide an OOBN representation of $\mathbb{F}$ through defining two other classes: the *Individual* ($\mathbb{I}$) and the *Segregation* ($\mathbb{S}$) class.

The individual class $\mathbb{I}$ is represented in Figure (6). If no information about the individual's parents is available, the allele input nodes A_0^i and A_1^i depend on the reference population parameters, otherwise they are determined by the transmitted alleles. Another input node is the binary random variable $\tilde{H}$ representing the originator status of a generic individual. To provide the transmission of the individual genetic characteristics to the child, a copy of the alleles is expressed as output nodes (A_0^o and A_1^o) and the other vertexes X and Z being interior nodes. The variable X denotes the observable genotype and its CPT is specified as follows

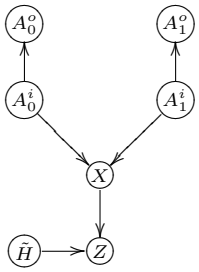


Figure 6: The individual class

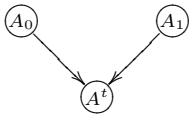


Figure 7: The segregation class

$$P(X = (a_r, a_u) \mid A_i^0 = a_h, A_i^1 = a_t) = \begin{cases} 1 & \text{if } (h = r \text{ and } t = u) \text{ or } (h = u \text{ and } t = r) \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

The segregation class, Figure (7), has two alleles as input nodes and provides the selection mechanism to generate the transmitted allele A^t via the following CPT, which reflects the first Mendelian law:

$$P(A^t = a_r \mid A_0 = a_t, A_1 = a_u) = \begin{cases} 1 & \text{if } r = t = u \\ 0.5 & \text{if } (r = t \text{ and } r \neq u) \text{ or } (r = u \text{ and } r \neq t) \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

On the whole the family class $\mathbb{F}$ is defined by a set of instantiations of $\mathbb{I}$, $\mathbb{I}(q)$, and $\mathbb{S}$, $\mathbb{S}(q,t)$, with $q, t \in \mathcal{F}$ and $q \neq t$. The index q is referred to the donor while t denotes the member who receives the allele in the segregation. The links among the instantiations of

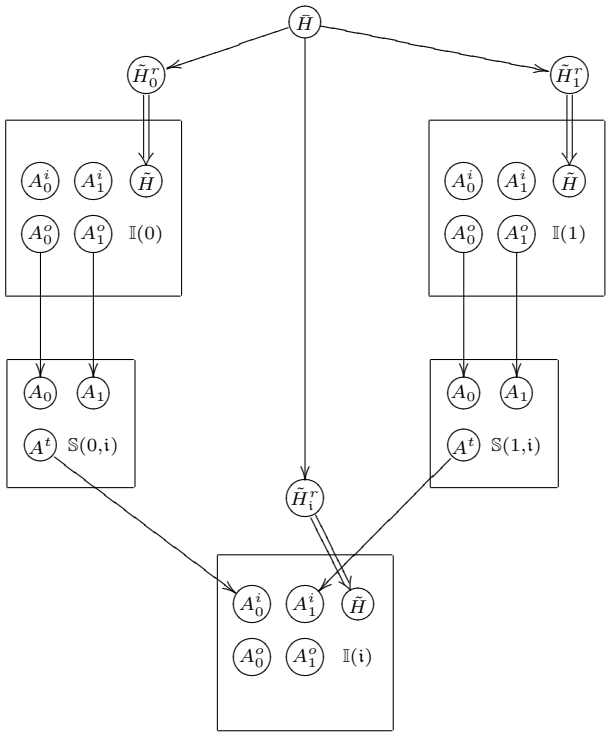


Figure 8: The family class $\mathbb{F}$ when $\mathcal{F} = \{0, 1, \mathrm{i}\}$.

the basic classes, $\mathbb{I}$ and $\mathbb{S}$, are drawn according to the biological relationships and each input node $\mathbb{I}(q).\tilde{H}$ has its own referenced vertex $\tilde{H}_q^r$. All of them are mixed by the input node $\bar{H}$ and the related CPTs are built as follows

$$P(\tilde{H}_q^r = 1 \mid \bar{H} = u) = \begin{cases} 1 & \text{if } q = u \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

with $u \in \bar{\mathcal{F}}$ and $q \in \mathcal{F}$. In Figure (8) we give a representation of $\mathbb{F}$, assuming, to simplify the picture, that $\mathcal{F} = \{0, 1, \mathrm{i}\}$.

The OOBN specified above deals with a single specific locus and it aims at the evaluation of the marginal posteriors for all the identification hypotheses.

In forensic practice, about 13-15 loci are usually typed for each individual and the support to the hypotheses is required conditionally to all the evidence.

Fortunately, this evaluation can be performed by using the results of the locus-specific nets, since linkage equilibrium still holds conditionally to the crime sample and the identification hypotheses. In fact, given an individual, the genotype distribution in a specific locus assumes the value of the genotype observed on the crime sample with probability one if identification is assumed; otherwise it follows the reference population distribution i.e. it never depends on the genotypes observed on other loci.

To give details, define: $\mathbb{L} = \{1, 2, \ldots, k\}$ the set of the loci; $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ the genotypes observed on the DB members and $\mathbf{x}_c = \{x_{c,1}, \ldots, x_{c,k}\}$ the crime samples observed on the considered loci.

If linkage equilibrium holds, the posterior of the identification hypothesis expressed in odds form, concerning, e.g., the q-th individual of the j-th family is:

$$\frac{P(H_j^r = q \mid \mathbf{x}, \mathbf{x_c})}{P(H_j^r \neq q \mid \mathbf{x}, \mathbf{x_c})} = \prod_{i=1}^{k} \frac{P(\mathbf{x}_i \mid x_{c,i}, H_j^r = q)}{P(\mathbf{x}_i \mid x_{c,i}, H_j^r \neq q)} \cdot \frac{P(H_j^r = q)}{P(H_j^r \neq q)}, \tag{14}$$

since $\forall i$, $P(x_{c,i} \mid H_j^r = q) = P(x_{c,i} \mid H_j^r \neq q)$. The first term on the RHS of (14) is the likelihood ratio (LR) and can be evaluated making use of the results provided by each locus-specific net after propagation, in fact, $\forall i \in \mathbb{L}$

$$\frac{P(\mathbf{x}_i \mid x_{c,i}, H_j^r = q)}{P(\mathbf{x}_i \mid x_{c,i}, H_j^r \neq q)} = \frac{P(H_j^r = q)}{P(H_j^r \neq q)} \cdot \frac{P(H_j^r \neq q \mid \mathbf{x}_i, x_{c,i})}{P(H_j^r = q \mid \mathbf{x}_i, x_{c,i})}. \tag{15}$$

Merits and difficulties to provide results as posteriors or LRs are discussed below.

1) The posterior probability of the hypothesis directly provides an answer to the uncertainty about the origin of the crime sample. Since a posterior requires the elicitation of a prior, this forces to deeply understand the meaning of each hypothesis, avoiding misunderstanding. This is a real problem as reveals the controversy between Stockmarr (1999), Dawid (2001) and Meester and Sjerps (2003): there the problem concerned the choice among hypotheses that sound logical. In this work both positions are represented: the Stockmarr's hypothesis is represented by the event $H \neq \mathtt{r}$ and considers the presence of the originator of the trace in the (augmented) DB; The Dawid's individual hypotheses are represented by the set of the H_j^rs. A possible drawback in the use of the posterior is that a large population size often implies very small (marginal) priors for each of the identification hypotheses so that small posteriors are likely to be obtained, wrongly suggesting a failure of the identification trial.

2) The LR is the measure usually provided to evaluate the evidence in a court; it does not imply any choice about priors and can be combined by the judge with others LRs obtained using different sources of evidence. An LR typically emphasizes a *discover*, even

Table 1: The rank distributions of the LR supporting the correct identification hypothesis.

Rank	Child	Brother
1°	54.99%	61.89%
2°	16.24%	10.71%
3°	7.53%	4.26%
4°	4.17%	2.36%
5°	2.90%	2.08%
6°	1.81%	1.27%
7°	1.63%	1.45%
$\leq 8°$	10.73%	15.98%

if the result might be of difficult interpretation, since the LR is not expressed in a normalized form.

4 APPLICATIONS

Now let us give account of a simulation study on a real DB containing 1102 observations on 10 loci. What is involved is how effective is the DB search in retrieving the origin of the simulated crime samples.

To produce the first simulation, we generated for each observed individual two crime samples obtained respectively sampling from the posterior marginal distribution of the child's and brother's genotypes. We call them the Child Crime Sample (CCS) and Brother Crime Sample (BCS).

Consider first the CCS. For each considered first-degree-relative pedigree we evaluate the hypothesis concerning the identification of the family originating the child. Obviously we expect that the LRs, or the posteriors, evaluated for the family from which the CCS was generated has one of the highest values. Similar computations are provided if the BCSs are used, and the results are in Table (1).

Concerning the identification of a child, in over 85% of the cases, the LR corresponding to the originating family ranks in the top five highest positions; the identification of a *brother* is slightly less successful, since in this case the same figure is just over 80%. In real cases, it seems safe to suggest that the the results' evaluation should include a comparison between the LRs or the posteriors for the families exhibiting the highest values associated to a careful investigative work.

As a comment, it must be noted that our simulation is disadvantaged with respect to real cases. For instance, when we sample a BCS we do not know the relatives' genotypes as the nature knows but our knowledge is restricted to the brother posterior distribution, typically over-dispersed. In real cases, brothers' genotypes are often very similar: for each locus, if one of the parents is homozygote the probability that brothers share one allele is equal to one and the probability they are identical is equal to 0.5.

Table 2: Parameter estimates of the CPU time proposed model

CPU	α	β	θ
Pentium IV	-10.93	1.82	0.01
AMD64	-11.75	1.84	0.01

A further simulation experiment has been achieved, making use of different DB sizes in the range $5000 - 50000$, and loci with a number of alleles varying in the range $5 - 20$. We estimate the dependence of the CPU times (t) required to perform the search with respect to the DB size (n) and the alleles' number (a) according to the model $\log(t) = \alpha + \beta \cdot \log(n) + \theta \cdot \log(a) + e$ where e is the stochastic error with zero mean. Results are in Table (2).

Clearly the estimation of the βs and the θs produced very similar results and the difference in technology is provided by α. Note that there is a very slight dependence on the number of the alleles, due to the adoption of an allele recoding strategy (Lauritzen and Sheehan, 2003). Instead the dependence of t on the DB size is less then quadratic, making the search feasible also when very large DB are involved.

5 CONCLUSIONS

The use of BN to provide an evaluation of the LR for forensic identification purposes is a new but already well-established approach, see Dawid (2003), Mortera and al. (2003) and Corradi et al. (2003).

Here, the BN technology is invoked when there is no clue about the origin of the trace, but a list of well identified individuals, not apparently related to the crime, is available in the DB. This result is all the more effective when an augmented DB is introduced, having assumed that all its members belong to the population of the crime sample's possible donors, even if some of them are not observed. In this new perspective the OOBN approach provides the most striking solution: the *familiar*, the *individual* and the *segregation* classes of hierarchy provide a concise representation of the repetitive part of the problem, saving efforts when *maintenance* operations are required. This could happen, for instance, when we want to introduce the possibility of a mutation in the alleles transmission: in this case a slight modification of the segregation class produces the result. At the same time the proposed solution leaves some room to operate on the single instance of the classes. This is compulsory for our problem since we are required not to consider as possible

originator of the crime sample those individuals in the augmented DB who are not included in the donors' population since e.g. dead or in jail. In the OOBN environment this can be realized just by intervening on the hypotheses input nodes concerning each family and detailed for each considered members.

PROOF OF PROPOSITION 2.1

The joint marginal distribution of $\{\mathbf{X}, \bar{\mathbf{H}}, H\}$ is the same in the two BNs of Figure (2) and Figure (3) so (8) becomes

$$\hat{P}(X_c = x, \mid \mathbf{X}, \bar{\mathbf{H}}) = C(x) \cdot \sum_{\mathbf{Z}} \tilde{P}(X_c^\star = x, \mid \mathbf{Z}) \cdot \prod_{j=1}^{n} \tilde{P}(Z_j \mid X_j, \bar{H}_j). \quad (16)$$

When the variable $X_c^\star$ receives an evidence $x \in \mathcal{X}$ it is easy to show that after the reduction (7) can be written as product of n potential ϕ_j, that is

$$\hat{P}(X_c^\star = x \mid \mathbf{Z}) = \prod_{j=1}^{n} \phi_j(Z_j) \quad (17)$$

where

$$\phi_j(Z_j = \hat{x}) = \begin{cases} 1 & \text{if } \hat{x} = x \\ 0 & \text{otherwise} \end{cases} \quad (18)$$

with $\hat{x} \in \mathcal{X}$.

The equation (18), which defines a *finding* on Z_j, establishes that all mediating variables take value x with probability 1. So, combining equations (17) and (18) with (16) we obtain

$$\hat{P}(X_c = x, \mid \mathbf{X}, \bar{\mathbf{H}}) = C(x) \cdot \prod_{j=1}^{n} \tilde{P}(Z_j = x \mid X_j, \bar{H}_j). \quad (19)$$

If $\bar{\mathbf{H}} = \mathbf{1}_j$ then from (2) and (4) we have

$$P(X_c = x, \mid X_j, H = j) = C(x) \cdot \prod_{i \neq j} \tilde{P}(Z_i = x \mid \bar{H}_i = 0) \cdot \tilde{P}(Z_j = x \mid X_j, \bar{H}_j = 1). \quad (20)$$

The third part of RHS of (20) involves $n-1$ terms. From (5), each of them is equal to θ_x so, considering (6) and assumption (*iii*) we obtain $C(x) = \theta_x^{1-n}$.

The same result is achieved for $\bar{\mathbf{H}} = \mathbf{0}$ as well. In fact, in that case, considering (2) and (4), the equation (19) becomes

$$P(X_c = x, \mid H = \mathtt{r}) = C(x) \cdot \prod_{j=1}^{n} \tilde{P}(Z_j = x \mid \bar{H}_j = 0). \quad (21)$$

Finally, from condition (*iv*) and equation (5) we obtain again $C(x) = \theta_x^{1-n}$.

REFERENCES

O. Bangso and P-H. Wuillemin (2000). Top-down Construction and Repetitive Structures Representation in Bayesian Networks. In *Proceedings of the Thirteenth International Florida Artificial Intelligence Society Conference*, 282-286. AAAI Press.

F. Corradi and G. Lago and F. M. Stefanini (2003). The Evaluation of DNA Evidence in Pedigrees Requiring Population Inference. *Journal of the Royal Statistical Society*, A166, 425-440.

A. P. Dawid (2001). Comment on Stockmarr's Likelihood Ratios for Evaluating DNA Evidence When the Suspect is Found Through a Database Search. In *Biometrics*, 57, 976-980.

A. P. Dawid (2003). An Object Oriented Bayesian Network for Estimating Mutation Rates. In *Proceeedings of the Ninth International Workshop on Artificial Intelligence and Statistics, January 3-6 2003, Key West, Florida*, edited by Christopher M. Bishop and Brendan J. Frey.

P. Donnelly and R.D. Friedman (1999). DNA Database Searches and the Legal Consumption of Science Evidence. *Michigan Law Review*, 974, 931-984.

D. Geiger and D. Heckerman (1996). Knowledge Representation and Inference in Similitary Networks and Bayesian Multinets. *Artificial Intelligence*, 82, 45-74.

F.V. Jensen (2001).Bayesian Network and Decision Graphs. *Springer-Verlag*, New York.

D. Koller and A. Pfeffer (1997). Object-Oriented Bayesian Network. *Proceedings of Thirteenth Conference on Uncertainty in Artificial Intelligence*, 302-313.

S. L. Lauritzen and N. A. Sheehan (2003). Graphical models for genetic analyses. *Statistical Science*, 18, 489-514.

R. Meester and M. Sjerps (2003). The Evidential Value in the DNA Database Search Controversy and the Two Stain Problem *Biometrics*, 59, 727-732

J. Mortera and A. P. Dawid and S. L. Lauritzen (2003). Probabilistic expert system for DNA mixture profiling. *Theoretical Population Biology*, 63, 191-205.

A. Stockmarr (1999). Likelihood Ratios for Evaluating DNA Evidence When the Suspect is Found Through a Database Search *Biometrics*, 55, 671-677

Active Learning for Parzen Window Classifier

Olivier Chapelle
Max Planck Institute for Biological Cybernetics
Spemannstr 38, 72076 Tübingen, Germany
olivier.chapelle@tuebingen.mpg.de

Abstract

The problem of active learning is approached in this paper by minimizing directly an estimate of the expected test error. The main difficulty in this "optimal" strategy is that output probabilities need to be estimated accurately. We suggest here different methods for estimating those efficiently. In this context, the Parzen window classifier is considered because it is both simple and probabilistic. The analysis of experimental results highlights that regularization is a key ingredient for this strategy.

1 Introduction

In the standard supervised framework, the goal is to estimate a function based on a given training set. *Active learning* is an extension of this framework where the learning machine does not only receive the training points passively, but can also choose the points to be included in the training set. An active learner may start with a small training set and at each iteration carefully selects one or several points for which it asks the labels to a human expert.

The main motivation for active learning is that it usually requires time and/or money for the human expert to label an example and those resources should not be wasted to label non-informative samples, but be spent on interesting ones.

Optimal Experimental Design (Fedorov, 1972) is closely related to active learning as it attempts to find a set of points such that the variance of the estimate is minimized. In contrast to this "batch" formulation, the term *active learning* often refers to an incremental strategy (Roy & McCallum, 2001; Sugiyama & Ogawa, 2000; Cohn et al., 1995; Sung & Niyogi, 1995; MacKay, 1992b).

We will concentrate on *pool-based* active learning (also called *selective sampling*): the learner can only query the labels of some points which belong to a large unlabeled set. Note that in this standard definition of pool-based active learning, the search is greedy: at each iteration, the goal is to find *one* point which will result in the smallest expected generalization error when added to the training set.

There has been various heuristic proposed for active learning, such as *uncertainty sampling* (Lewis & Gale, 1994) or *version space minimization* (Freund et al., 1997; Tong & Koller, 2001). However, ideally, the aim is to choose the point such that the expected test error is minimized (Roy & McCallum, 2001). Such an approach has also been suggested in (Schohn & Cohn, 2000) in the context of *Support Vector Machines* learning, but the authors argued that it would be computationally intractable.

However this "optimal" approach has been implemented for SVMs and Parzen window classifier in (Chapelle, 2003, chapter 8), but it performed terribly. In this paper, we investigate why the naive implementation of this active learning strategy does not works and suggest two remedies based on semi-supervised learning and regularization.

The paper is organized as follows: section 2 presents the strategy which consists in minimizing the expected test error and section 3 shows how to apply it to the Parzen window classifier. One of the reason for considering this simple classifier is that a more sophisticated classifier might introduce a bias in our analysis of active learning. In section 4, we propose a first improvement that takes into account the unlabeled points for the class conditional density estimates, and finally, section 5 introduces approaches based on regularization.

Note that for convenience, experimental results will be presented all along the paper in order to assess immediately the performance of a new method.

2 Optimal active learning

The optimal active learning strategy we present here has been described for instance in (Schohn & Cohn, 2000; Roy & McCallum, 2001). It consists in querying the label of the point, that once incorporated in the training set, will yield the lowest expected test error.

Let $D = (\mathbf{x}_i, y_i)_{1 \leq i \leq n}$ be the training samples. Suppose that the generalization error of the function learned on this training set can be estimated. Let us denote by $T(D)$ such an estimate (which, of course, depends also on the learning algorithm). The optimal active learning strategy would be the following,

1. Train the classifier using the current training set of n points and get the hypothesis $\hat{f}_n$.
2. Fix a point $\mathbf{x}$ in the unlabeled set
   (a) Fix a label y and add the point $(\mathbf{x}, y)$ in the training set
   (b) Retrain the classifier with the additional point.
   (c) Estimate the generalization error $T(D \cup (\mathbf{x}, y))$.
   (d) Estimate the posterior probability $\hat{P}(y|\mathbf{x}, \hat{f}_n)$ of the new point under the hypothesis $\hat{f}_n$.
   (e) Compute the expected generalization error $\bar{T}_{\mathbf{x}} = \sum_y \hat{P}(y|\mathbf{x}, \hat{f}_n) T(D \cup (\mathbf{x}, y))$.
3. Choose for labeling the point $\mathbf{x}$ which has the lowest expected generalization error $\bar{T}_{\mathbf{x}}$ and add it to training set.

There are several problems with this strategy. Beside computational difficulties (at a first sight, a lot of retrainings are necessary), the fundamental problem is how to compute T and $\hat{P}(y|\mathbf{x}, \hat{f}_n)$. Note that in the rest of the paper, we will refer indifferently to $\hat{P}(y|\mathbf{x})$ as posterior or output probability.

For classification, the posterior probability can be used directly (Roy & McCallum, 2001; Zhu et al., 2003) to compute T,

$$T = \frac{1}{n_u} \sum_{i=n+1}^{N} \left(1 - \max_{y \in \{-1,1\}} P(y|\mathbf{x}_i, \hat{f}_n)\right), \quad (1)$$

where $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N$ is the set of unlabeled data and $P(y|\mathbf{x}, \hat{f})$ is an estimate of the posterior probability for the point $\mathbf{x}$ given the function $\hat{f}_n$ learned on the training set.

Eq. (1) can be seen as the empirical counter part of

$$\frac{1}{2} \iint |y - \arg\max P(y|\mathbf{x}_i, \hat{f}_n)| \;\; dP(y|\mathbf{x}, \hat{f}_n) dP(\mathbf{x}),$$

which would be the the generalization error if $dP(y|\mathbf{x}, \hat{f}_n)$ were the true conditional distribution of y given $\mathbf{x}$.

3 Parzen window classifier

The goal is to apply this strategy with the Parzen window classifier. In this case, we have

$$\hat{P}(\mathbf{x}|y) = \frac{1}{|\{i|\ y_i = y\}|} \sum_{i,\ y_i = y} K(\mathbf{x}, \mathbf{x}_i) \quad (2)$$

and by Bayes rule

$$\hat{P}(y|\mathbf{x}) = \frac{\sum_{i,\ y_i = y} K(\mathbf{x}, \mathbf{x}_i)}{\sum_{i=1}^{n} K(\mathbf{x}, \mathbf{x}_i)}, \quad (3)$$

where K is typically a Gaussian kernel of the form (up to an irrelevant multiplicative constant),

$$K(\mathbf{x}, \mathbf{x}') = \exp(-||\mathbf{x} - \mathbf{x}'||^2 / 2\sigma^2).$$

It is thus possible to compute the estimated generalization error using (1) and perform the optimal active learning strategy described above. The reason for considering this simple classifier with active learning is that equation (3) gives directly an estimate of the posterior probability and that it does need a costly retraining when a point is added to the training set.

Experiments

Two datasets have been used for the experimental results presented in this paper: an artificial one and a real world one, and the details of the experimental setup are as indicated below.

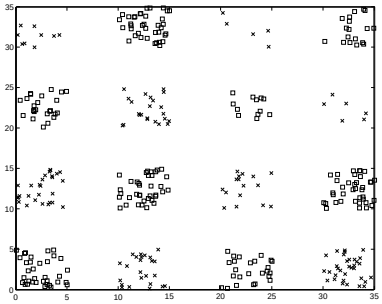


Figure 2: Toy problem: checker board dataset

Toy problem This is a modified version of the toy problem used in (Zhu et al., 2003). As plotted in figure 2, it consists of a checker board, where in each cluster, the points are drawn according to a uniform distribution and the number of point is

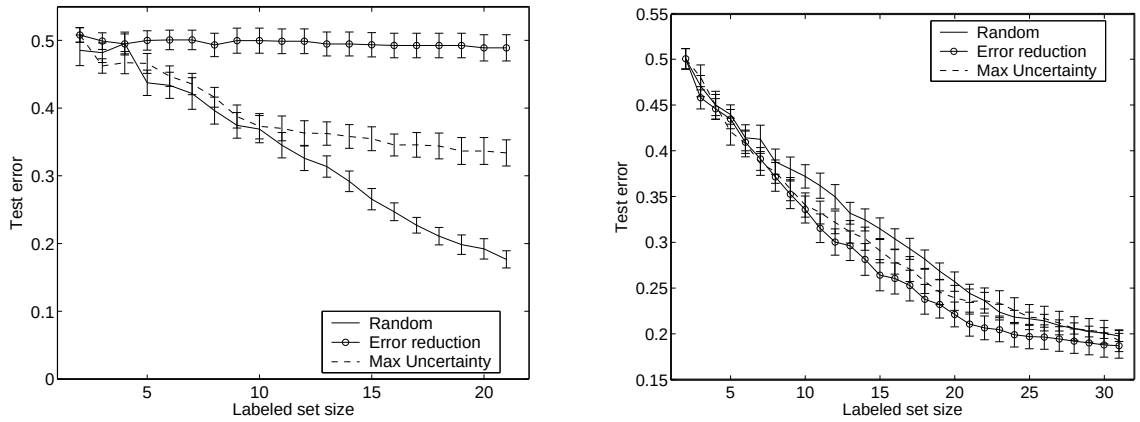


Figure 1: Test errors achieved by 3 active learning strategies on the toy problem (left) and the USPS database (right)

also drawn randomly between 1 and 40. 2 labeled points (one negative and one positive) are selected randomly and 20 samples are chosen incrementally to be labeled. Test errors are computed on the non-queried points and averaged over 100 trials. As in (Zhu et al., 2003), the variance of kernel estimate was fixed at $\sigma^2 = 2$.

Digit classification The real world database is the USPS one consisting of 7291 training samples and 2007 test ones. The training samples have been divided on 23 subsets of 317 examples each. The task is to classify digits 0 to 4 against 5 to 9. As for the toy problem, 2 random labeled points are selected and 30 samples among the 315 remaining are queried for their labels. The width of the Parzen classifier was set to $\sigma^2 = 256 \cdot (0.1)^2$, which gave the best performance in a standard supervised framework. Note that this is actually a very small value and the resulting classifier behaves almost as 1-nearest neighbor classifier.

Experimental results are provided in figure 1, where the method described in this section (entitled error reduction) is compared to random queries and to the standard max uncertainty strategy which selects the point $\mathbf{x}_i$ for which the learner is the most uncertain, i.e. whose output probability (3) is the nearest from 1/2.

The results for the error reduction strategy are really disappointing: it is not much better than random on USPS and is terrible on the toy problem.

The conjecture of why it failed is because the strategy presented in the previous section depends heavily on reliable estimates of the posterior probabilities (through the step (e) and equation (1)). For this reason, we will try in the rest of the paper to have more reliable estimates, but note that in most cases, the decision function given by $\arg\max \hat{P}(y|\mathbf{x})$ will remain unchanged (except in section 4.2).

A first observation which shows that the density estimates are not very reliable is the following: for a given point $\mathbf{x}$, the value of $P(\mathbf{x})$ can either be estimated as $1/N \sum_{i=1}^{N} K(\mathbf{x}, \mathbf{x}_i)$ [Parzen window on all the points] or as $\sum_y \hat{P}(\mathbf{x}|y)\hat{P}(y) = 1/n \sum_{i=1}^{n} K(\mathbf{x}, \mathbf{x}_i)$ [Parzen window on the labeled points]; and those two values can be quite different.

4 Class conditional density estimate using unlabeled points

As mentioned above, the standard Parzen window estimator of the class conditional densities does not take into account the unlabeled points. In other words,

$$\hat{P}(\mathbf{x}_i, y_i = 1) + \hat{P}(\mathbf{x}_i, y_i = -1) \neq \tilde{P}(\mathbf{x}_i), \qquad (4)$$

where $\tilde{P}$ is the Parzen window estimator on the labeled and unlabeled points.

We will discuss two ways to solve the "contradiction" revealed by equation (4).

4.1 Constrained Parzen window

The first idea is not to estimate both class conditional densities independently, but in such a way that equality (4) holds. In (Vapnik, 1998), it was shown that the Parzen window estimator can be seen as a solution of an optimization problem consisting of a smoothness term and a term fitting the data (the L_2 error between the empirical distribution function and the estimated one).

Based on this observation, it was suggested in (Chapelle, 2003, Chapter 7) in the context of semi-supervised learning to explicitly add equality (4) as a constraint in the optimization problem and this constrained Parzen window estimate turns out to be

$$\hat{P}_{CTR}(\mathbf{x}_i, y_i = 1) = \hat{P}(\mathbf{x}_i, y_i = 1) + \frac{1}{2}\Delta P(\mathbf{x}_i), \quad (5)$$

where ΔP is the difference between the left and right hand side of (4), i.e. $\Delta P(\mathbf{x}) = \tilde{P}(\mathbf{x}) - \sum_y \hat{P}(\mathbf{x}_i, y_i = y)$ With this new estimate of class conditional density, (4) becomes now an equality.

However, this modification raises another problem: when $\Delta P < 0$, it might happen that the output probability $\hat{P}_{CTR}(y|\mathbf{x})$ is no longer between 0 and 1. In this case, we decided to threshold the output to 0 or 1.

A middle way solution is to add only a fraction of ΔP, i.e. to replace equation (5) by,

$$\hat{P}_{CTR}(\mathbf{x}_i, y_i = 1) = \hat{P}(\mathbf{x}_i, y_i = 1) + \frac{\gamma}{2}\Delta P(\mathbf{x}_i), \quad (6)$$

where γ is chosen between 0 and 1.

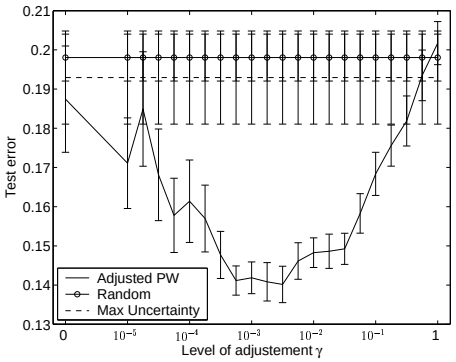


Figure 3: Test error as a function of γ in (6) after 30 points were added in the labeled set of USPS.

As plotted in figure 3, there can be a very significant improvement when γ is chosen appropriately. However, it is not clear how it should be chosen. Also, the fact of having to threshold the output probabilities because they are not always between 0 and 1 is not very satisfactory. Future research includes the derivation an improved constrained Parzen window estimate.

4.2 Expansion on the unlabeled points

A second idea is to use all the points in the expansion of the class distribution (2). First, suppose that the labels of the unlabeled points were known. Then, the class Parzen window estimate would give

$$\hat{P}(x, y|y_{n+1,\ldots,N}) = \frac{1}{N} \sum_{i=1,\ y_i=y}^{N} K(\mathbf{x}, \mathbf{x}_i).$$

Introducing the variables $\lambda_i = P(y_i = 1|\mathbf{x}_i)$ and integrating over the choice of the unknown labels of the unlabeled points, we then have

$$\hat{P}(\mathbf{x}, y = 1) = \frac{1}{N} \sum_{i=1}^{N} \lambda_i K(\mathbf{x}, \mathbf{x}_i). \quad (7)$$

The λ_i for the unlabeled points are of course unknown, but we will see how to estimate them. The λ_i for the labeled points are set in this section to 0 or 1, according to the labels y_i, i.e. $\lambda_i = (y_i + 1)/2$.

Now note that by conditioning on $\mathbf{x}$ equation (7) gives the conditional probability output of a point under the Parzen window model,

$$\hat{P}(y_p = 1|\mathbf{x}_p) = \frac{\sum_{i=1}^{N} \lambda_i K(\mathbf{x}_i, \mathbf{x}_p)}{\sum_{i=1}^{N} K(\mathbf{x}_i, \mathbf{x}_p)} \equiv \tilde{\lambda}_p,$$

that we can rewrite in matrix notation as

$$\tilde{\boldsymbol{\lambda}} \equiv D^{-1} K \boldsymbol{\lambda}, \quad (8)$$

where D is a diagonal matrix with $D_{ii} = \sum_j K_{ij}$ and $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$.

A way to estimate $\boldsymbol{\lambda}$ is to enforce that $\lambda_i = \tilde{\lambda}_i$ for each unlabeled point $\mathbf{x}_i$. By doing so, the model is coherent. Splitting equation (8) between labeled and unlabeled blocks, this constraint writes

$$\boldsymbol{\lambda}_u = (D^{-1}K)_{u,u}\boldsymbol{\lambda}_u + (D^{-1}K)_{u,l}\boldsymbol{\lambda}_l,$$

where the subscripts l and u stand respectively for the labeled and unlabeled indices. And since $\boldsymbol{\lambda}_l = (Y + 1)/2$, we get

$$\begin{aligned} \boldsymbol{\lambda}_u &= (I - (D^{-1}K)_{u,u})^{-1}(D^{-1}K)_{u,l}(Y + 1)/2 \\ &= [(D - K)_{u,u}]^{-1} K_{u,l}(Y + 1)/2, \end{aligned}$$

which is exactly how the output probabilities are estimated in (Zhu et al., 2003).

This way of estimating of the output probabilities yield directly an active learning algorithm once it is combined with the framework presented in section 2. This algorithm was suggested in (Zhu et al., 2003) and the experimental results therein are quite impressive. An explanation for these good performances is that there is a *semi-supervised* learning step in this algorithm. Indeed, consider an unlabeled point $\mathbf{x}_i$ for which $P(\mathbf{x}_i, y_i = 1) \approx P(\mathbf{x}_i, y_i = -1) \approx 0$. Then, the constrained Parzen window estimator will correct the class conditional density estimates by adding the same value $\Delta P(\mathbf{x}_i)/2$ to both of them, whereas the semi-supervised one will choose a value λ_i between 0 and 1 according to what is the most "likely" label of $\mathbf{x}_i$.

Figure 4 confirms that significant improvements are indeed obtained using this method (referred in the plot as semi-supervised).

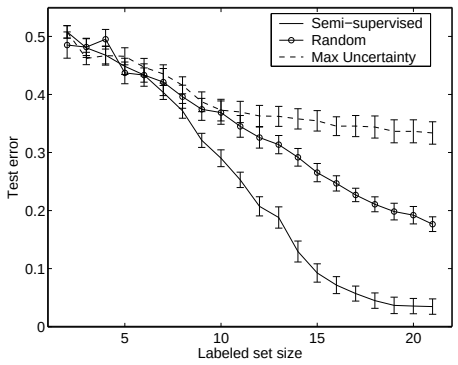


Figure 4: Results on the toy database for the method described in this section (semi-supervised), which was first introduced in (Zhu et al., 2003)

4.3 Soft margin formulation

We now consider the λ_i for the labeled points as free variables.

For a fixed value of the vector $\boldsymbol{\lambda}_l$, the posterior probabilities on the unlabeled points are given, as in the previous section by

$$\boldsymbol{\lambda}_u = (I - (D^{-1}K)_{u,u})^{-1}(D^{-1}K)_{u,l}\boldsymbol{\lambda}_l. \qquad (9)$$

We suggest to find $\boldsymbol{\lambda}_l$ as the solution of an optimization problem consisting of a likelihood term and a "coherence" term.

Firstly, conditioning on the inputs and on the output probabilities $\boldsymbol{\lambda}_l$, the log-likelihood of the labels is

$$\log P(y_{1..n}|\mathbf{x}_{1..n}, \boldsymbol{\lambda}_u) = \sum_{i=1}^{n} \frac{1+y_i}{2}\log\lambda_i + \frac{1-y_i}{2}\log(1-\lambda_i) \equiv L(\boldsymbol{\lambda}_l),$$

which should be maximized.

Secondly, note that both $\boldsymbol{\lambda}$ and $\tilde{\boldsymbol{\lambda}}$ are estimate of the posterior probabilities and ideally those two vectors should be identical. By definition of the choice of $\boldsymbol{\lambda}_u$ in (9), we have already $\boldsymbol{\lambda}_u = \tilde{\boldsymbol{\lambda}}_u$. Even though it is impossible to get $\boldsymbol{\lambda}_l = \tilde{\boldsymbol{\lambda}}_l$, one can try to minimize the difference between $\boldsymbol{\lambda}_l$ and $\tilde{\boldsymbol{\lambda}}_l$. Their discrepancy is somehow a measure of how incoherent the model is. Since both vectors are actually estimates about the conditional distribution, $P(y|\mathbf{x})$, it seems natural to use the Kullback-Leibler divergence, which is in this case, under an independence assumption,

$$KL(\boldsymbol{\lambda}_l, \tilde{\boldsymbol{\lambda}}_l) = \sum_{i=1}^{n} \lambda_i \log\left(\frac{\lambda_i}{\tilde{\lambda}_i}\right) + (1-\lambda_i)\log\left(\frac{1-\lambda_i}{1-\tilde{\lambda}_i}\right).$$

If $\boldsymbol{\lambda}_l$ and $\tilde{\boldsymbol{\lambda}}_l$ are close enough, a first order expansion gives

$$KL(\boldsymbol{\lambda}_l, \tilde{\boldsymbol{\lambda}}_l) \approx \sum_{i=1}^{n} \frac{(\lambda_i - \tilde{\lambda}_i)^2}{\lambda_i(1-\lambda_i)} = W(\boldsymbol{\lambda}_l, \boldsymbol{\lambda}_l - \tilde{\boldsymbol{\lambda}}_l),$$

where $W(\mathbf{x}, \mathbf{y}) \equiv \sum \frac{y_i^2}{x_i(1-x_i)}$.

Let us see how to compute $\lambda_i - \tilde{\lambda}_i$. For this purpose, we introduce $S = I - D^{-1}K$, and using block matrix identities as well as (9), we get

$$\begin{aligned} \lambda_i - \tilde{\lambda}_i &= \left[(I - D^{-1}K)\begin{pmatrix} \boldsymbol{\lambda}_l \\ \boldsymbol{\lambda}_u \end{pmatrix}\right]_i \\ &= [(S_{ll} - S_{lu}S_{uu}^{-1}S_{ul})\boldsymbol{\lambda}_l]_i \\ &= [(S^{-1})_{ll}^{-1}\boldsymbol{\lambda}_l]_i \end{aligned}$$

Putting everything together, we suggest to find the vector $\boldsymbol{\lambda}_l$ which minimizes

$$-\gamma L(\boldsymbol{\lambda}_l) + W(\boldsymbol{\lambda}_l, (S^{-1})_{ll}^{-1}\boldsymbol{\lambda}_l), \qquad (10)$$

and to get $\boldsymbol{\lambda}_u$ from $\boldsymbol{\lambda}_l$ through equation (9).

Note that the minimization of (10) is a convex optimization problem as shown in appendix.

As a side remark, one might be worried by the computational complexity of this method as well as some others presented in this paper. Indeed, at each iteration, and for each candidate, the λ_i need to be re-estimated (step (b) in section 2), which would be prohibitive if those updates were done naively. However, using rank-one updates and block matrix identities, one can compute efficiently the new gradients and Hessian of the objective function (10) when a labeled point is added. From there, a Newton's step is simulated in order to update $\boldsymbol{\lambda}_l$.

Experimental results using this soft margin formulation are presented in table 1 and on both databases it did not help. This might be because those datasets are not really noisy (for the postal dataset, a hard margin SVM performs better than a soft margin one). However, in future experiments, we will experiment this algorithm on noisy datasets.

	1	10	1000	Hard margin
Checker board	20.8	20.9	21.5	21.5
USPS	16.5	15.6	15.5	15.2

Table 1: Test error as a function of the soft margin parameter γ after 10 queried points for the checker board dataset and 30 for USPS

5 Uncertain posterior probabilities

Consider the example presented in figure 5, which is quite similar to the one in (Zhu et al., 2003) and pick

one point in the cluster on right hand side. Then the probability for this point to belong to either class is very small because it is far from both labeled points. However, since the point is nearer from the circle, the posterior probability estimated by Parzen window of its label being a circle is almost 1.

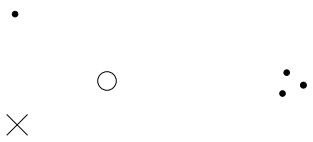


Figure 5: The unlabeled points on the right hand side will not be queried because they are (maybe wrongly) believed to be circles.

Intuitively, this is not very satisfactory: the active learning algorithm will not select a point from this cluster. This is one of the problem occurring in the approach presented above as well as in (Roy & McCallum, 2001; Zhu et al., 2003) is that it uses estimates of the posterior probabilities but ignores their variance or how reliable those estimates are.

5.1 Constrained Parzen window

The constrained Parzen window solves this problem as in regions of the space where there are unlabeled data which are far from the labeled ones, it increases the joint density estimates and the amount by which it is increased is the same for both classes. Thus in this case, the posterior probability is near from 1/2.

5.2 Regularizing

If there is an uncertainty about the posterior probability of a point, this one should be pushed towards 1/2. There are several possibilities for achieving this goal. First, let us introduce the log ratio of the posterior probabilities,

$$\alpha_i = \log \frac{P(y_i = 1|\mathbf{x}_i)}{P(y_i = -1|\mathbf{x}_i)}. \tag{11}$$

Quadric penalty The first idea is to introduce a quadratic regularization term on $\boldsymbol{\alpha}$ in the objective function we want to minimize. In this way, the α_i which are not constrained by some other terms will have small values and thus, the corresponding λ_i nearer from 1/2.

Using the variance Suppose that there is a Gaussian error on the value of $\boldsymbol{\alpha}$ and that we know its variance $\delta\boldsymbol{\alpha}$. Then MacKay (MacKay, 1992a) suggests to replace α_i by

$$\frac{\alpha_i}{\sqrt{1 + \pi(\delta\alpha)_i^2/8}}.$$

By doing so, if $(\delta\alpha)_i$ is small, the posterior probability is almost unchanged. However, if it is large, then the new α_i is small, which means that $P(y_i = 1|\mathbf{x}_i)$ is closer to 1/2. This is exactly the desired effect.

The variance can be estimated (up to a multiplicative constant) thanks to the Hessian H of the objective function at the optimal value, $(\delta\alpha)_i^2 \propto H_{ii}^{-1}$.

Regularized Parzen window When observing n^+ positive examples and n^- negative ones, the standard way to estimate the ratio of the positive class is to use a Beta prior on the class probability, which leads to the following estimate, $\frac{n^+ + 1}{n^+ + n^- + 2}$.

Based on this observation, one can estimate the posterior probability using the Parzen window estimate as,

$$\hat{P}(Y = 1|X = \mathbf{x}_p) = \frac{\sum \delta_{y_i=1} K(\mathbf{x}_i, \mathbf{x}_p) + \varepsilon}{\sum K(\mathbf{x}_i, \mathbf{x}_p) + 2\varepsilon}, \tag{12}$$

where ε is a small constant to be chosen.

Those three methods require a constant to be chosen, which represents the amount of regularization. We decided to the consider the last one because ε has a more direct interpretation in terms of prior probability. Also, it might be interesting to choose ε as to minimize the KL divergence between the density estimated on the unlabeled points, $\sum_{i=1}^{N} K(\mathbf{x}_i, \mathbf{x})$ and the "regularized" density on the labeled points, $\sum_{i=1}^{n} K(\mathbf{x}_i, \mathbf{x}) + 2\varepsilon$.

The results of the experiments presented in figure 6 show that this regularization is extremely useful, especially for the error reduction strategy which performed poorly without regularization (see section 3). The semi-supervised method described in section 4.2 behaves also much better with regularization. Note the local maximum in the right plot of figure 6. This is quite surprising and requires further investigation.

6 Conclusion

This paper provided an analysis of the influence of reliable posterior probabilities estimates on the performance of an active learner. In particular, it showed that regularization seems to be a very useful ingredient in those estimations.

Figure 7 shows that using this regularization, the performances achieved are not far from the best achievable ones in the case of the toy problem, and also for the USPS database after 30 queries.

The conclusions drawn from the analysis in this paper should be useful to adapt this active learning strat-

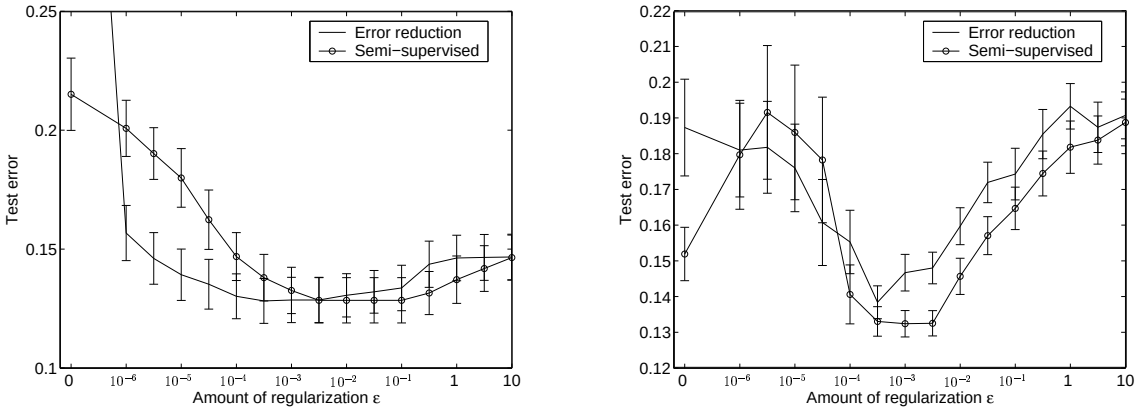


Figure 6: Experiments on the checker board dataset (left) and on USPS (right). The test error is plotted as a function of ε used to estimated the regularized posterior probability (12).

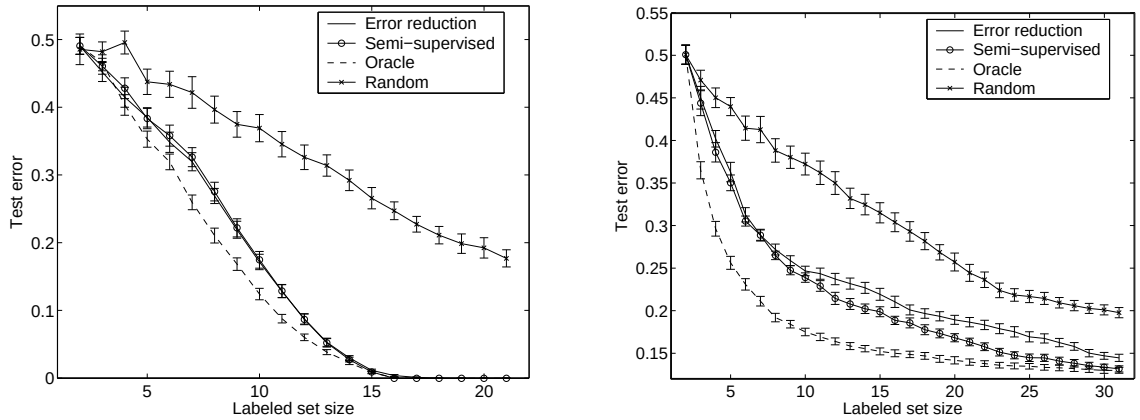


Figure 7: Performances achieved on the toy problem (left) and USPS (right). Both error reduction and semi-supervised use a regularization parameter $\varepsilon = 10^{-3}$ (see also figure 6). The Oracle strategy is the best one can achieve: it estimates the test error (1) using the *true* labels of the unlabeled set.

egy to more sophisticated classifiers such as Support Vector Machines or Gaussian Processes.

Appendix

The functional (10) is a convex function of $\boldsymbol{\lambda}_l$.

Indeed, computing the second derivatives of L, it is easy to check that L is a concave function. Concerning W, all the terms are of the form $\frac{(T\boldsymbol{\lambda})_i^2}{\lambda_i(1-\lambda_i)}$, which can be shown to be convex thanks to the following lemma

Lemma 1 *If f is a convex non-negative function on $\mathbb{R}^n$ and g is a concave positive function on $\mathbb{R}^n$, then f^2/g is convex.*

Proof: The Hessian of f^2/g is

$$\nabla^2 \frac{f^2}{g} = \frac{2f}{g}\nabla^2 f - \frac{f^2}{g^2}\nabla^2 g + \frac{2}{g^3}(g\nabla f - f\nabla g)(g\nabla f - f\nabla g)^\top,$$

which is a sum of positive definite matrices □.

We then apply the previous lemma with the convex function $f(\boldsymbol{\lambda}) = |\sum_j S_{ij}\lambda_j|$ (it satisfies Jensen's inequality) and the concave function $g(\boldsymbol{\lambda}) = \lambda_i(1-\lambda_i)$.

Instead of optimizing on $\boldsymbol{\lambda} \in [0,1]^n$, in practice we optimize on $\alpha_i = \log \lambda_i/(1-\lambda_i)$ (see also (11). This leads to an unconstrained optimization problem which is easier to solve numerically.

Note that by doing this change of variable, the objective function is not convex anymore. However, since

this is a monotonic transformation, it is easy to show that the function is quasiconvex (Boyd & Vandenberghe, 2003), and can thus be minimized efficiently (there is no local minima).

Acknowlegments

References

Boyd, S., & Vandenberghe, L. (2003). *Convex optimization.* Cambridge University Press.

Chapelle, O. (2003). *Support vector machines: Induction principle, adaptive tuning and prior knwoledge.* Doctoral dissertation, LIP6.

Cohn, D. A., Ghahramani, Z., & Jordan, M. I. (1995). Active learning with statistical models. *Advances in Neural Information Processing Systems* (pp. 705–712). The MIT Press.

Fedorov, V. (1972). *Theory of optimal experiments.* New York: Academic Press.

Freund, Y., Seung, H. S., Shamir, E., & Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, *28*, 133–168.

Lewis, D., & Gale, W. (1994). Training text classifiers by uncertainty sampling. *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 3–12).

MacKay, D. (1992a). The evidence framework applied to classification networks. *Neural Computation*, *4*, 720–736.

MacKay, D. (1992b). Information-based objective functions for active data selection. *Neural Computation*, *4*, 590–604.

Roy, N., & McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. *Proceedings of the International Conference on Machine Learning.*

Schohn, G., & Cohn, D. (2000). Less is more: Active learning with support vector machines. *Proceedings of 17th International Conference on Machine Learning* (pp. 839–846). San Francisco, CA: Morgan Kaufmann.

Sugiyama, M., & Ogawa, H. (2000). Incremental active learning for optimal generalization. *Neural Computation*, *12*, 2909–2940.

Sung, K. K., & Niyogi, P. (1995). Active learning for function approximation. *Advances in Neural Information Processing Systems* (pp. 593–600). The MIT Press.

Tong, S., & Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research* (pp. 45–66).

Vapnik, V. (1998). *Statistical Learning Theory.* John Wiley & Sons.

Zhu, X., Lafferty, J., & Ghahramani, Z. (2003). Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. *ICML workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining.*

Semi-Supervised Classification by Low Density Separation

Olivier Chapelle, Alexander Zien
Max Planck Institute for Biological Cybernetics
72076 Tübingen, Germany

Abstract

We believe that the cluster assumption is key to successful semi-supervised learning. Based on this, we propose three semi-supervised algorithms: 1. deriving graph-based distances that emphazise low density regions between clusters, followed by training a standard SVM; 2. optimizing the Transductive SVM objective function, which places the decision boundary in low density regions, by gradient descent; 3. combining the first two to make maximum use of the cluster assumption. We compare with state of the art algorithms and demonstrate superior accuracy for the latter two methods.

1 INTRODUCTION

The goal of semi-supervised classification is to use unlabeled data to improve the generalization. The *cluster assumption* states that the decision boundary should not cross high density regions, but instead lie in low density regions. We believe that virtually all successful semi-supervised algorithms utilize the cluster assumption, though most of the time indirectly.

For instance, manifold learning algorithms (e.g., [1]) construct decision functions that vary little along the manifolds occupied by the data. Often, different classes form separate manifolds. Then, manifold learning indirectly implements the cluster assumption by not cutting the manifolds.

The Transductive SVM [20] implements the cluster assumption more directly by trying to find a hyperplane which is far away from the unlabeled points. In our opinion, the rationale for maximizing the margin is very different for the labeled and unlabeled points:

- For the labeled points, it implements regularization [20]. Intuitively, the large margin property makes the classification robust with respect to perturbations of the data points [6].
- For the unlabeled points, the margin maximization implements the cluster assumption. It is not directly related to regularization (in this respect, we have a different view than Vapnik [20]). Consider for instance an example where the cluster assumption does not hold: a uniform distribution of input points. Then the unlabeled points convey almost no information, and maximizing the margin on those points is useless (and can even be harmful).

TSVM might seem to be the perfect semi-supervised algorithm, since it combines the powerful regularization of SVMs with a direct implementation of the cluster assumption. However, its main drawback is that the objective function is non-convex and thus difficult to minimize. Consequently, optimization heuristics like `SVMlight` [12] sometimes give bad results and are often criticized. The main points of this paper are:

- The objective function of TSVM is appropriate, but different ways of optimizing it can lead to very different results. Thus, it is more accurate to criticize a given implementation of the TSVM rather than the objective function itself.
- The search for a low density decision boundary is difficult. The task of the TSVM algorithm can be eased by changing the data representation.

To substantiate our claims, we develop and assess corresponding algorithms. Firstly, we propose a graph-based semi-supervised learning method exploiting the cluster assumption. Secondly, it is shown that a gradient descent on the primal formulation of the TSVM objective function performs significantly better then the optimization strategy pursued in `SVMlight` [12]. Finally, by combining these two ideas in one algorithm, we are able to achieve clearly superior generalization accuracy.

2 ALGORITHMS

Let the given data consist of n labeled data points $\mathbf{x}_i, 1 \le i \le n$, and m unlabeled data points $\mathbf{x}_i, n+1 \le i \le n+m$. For simplicity, we assume that the labels $y_i, 1 \le i \le n$, are binary, i.e. $y_i = \pm 1$; for multi-class problems, we use the one-against-rest scheme that is common for SVMs (e.g., [17]).

In the following sections, we describe two different ways to enforce the cluster assumption in SVM classification and how they can be implemented.

2.1 GRAPH-BASED SIMILARITIES

Let the graph $\mathsf{G} = (V, E)$ be derived from the data such that the nodes are the data points, $V = \{\mathbf{x}_i\}$. If sparsity is desired, edges are placed between nodes that are nearest neighbors (NN), either thresholding the degree (k-NN)[1] or the distance (ϵ-NN). Many semi-supervised learning methods operate on nearest neighbor graphs, see e.g. [1, 14, 18, 23, 22]. Usually they do not require the data points themselves, but only their pairwise distances along the edges. In the following we assume that the edges $(i, j) \in E$ are weighted by Euclidean distances $d(i, j) := ||\mathbf{x}_i - \mathbf{x}_j||_2$ (missing edges correspond to $d(i, j) = \infty$), although other distances are possible as well.

Many graph-based semi-supervised algorithms work by enforcing smoothness of the solution with respect to the graph, i.e. that the output function varies little between connected nodes. Here we use the graph to derive pairwise similarities between points, thereby "squeezing" the distances in high density regions while leaving them in low density regions. This idea has been proposed before, e.g. in [5, 21] and [4, section 3]. It has been implemented and used in `Isomap` [19], cluster kernels [7], and connectivity clustering [10].

2.1.1 Motivation

According to the cluster assumption, the decision boundary should preferably not cut clusters. A way to enforce this for similarity-based classifiers is to assign low similarities to pairs of points that lie in different clusters. To do so, we construct a Parzen window density estimate with a Gaussian kernel of width $\frac{1}{\sqrt{2}}\sigma$,

$$\hat{p}(\mathbf{x}') \quad = \quad \frac{1}{\sqrt{\pi}\sigma} \sum_{i=1}^{n+m} \exp\left(-\frac{||\mathbf{x}' - \mathbf{x}_i||^2}{\sigma^2}\right).$$

If two points are in the same cluster, it means that there exists a continuous connecting curve that only goes through regions of high density; if two points are in different clusters, every such curve has to traverse a density valley. We can thus define the similarity of two points by maximizing over all continuous connecting curves the minimum density along the connection, but this is hard to compute.

Two observations, illustrated in Figure 1, allow to approximate the above similarity with paths on a graph: (a) An optimal connecting curve can be well approximated by conjoining short line segments that directly connect points. (b) The minimum density is assumed at the middle of a line segment, and dominated by the closest points.

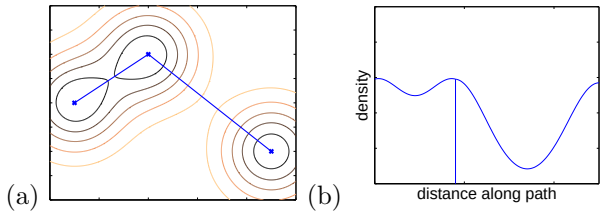


Figure 1: Optimal connecting curves are well approximated by paths of short distance edges on a graph.

2.1.2 A density-sensitive distance measure

Formally, we define $\mathsf{p} \in V^l$ to be a path of length $l =: |\mathsf{p}|$ on a graph $\mathsf{G} = (V, E)$, if $(p_k, p_{k+1}) \in E$ for $1 \le k < |\mathsf{p}|$. A path p is said to connect the nodes p_1 and $\mathsf{p}_{|\mathsf{p}|}$; let $\mathsf{P}_{i,j}$ denote the set of all paths connecting $\mathbf{x}_i$ and $\mathbf{x}_j$. We obtain

$$\begin{aligned} &\max_{\mathsf{p} \in \mathsf{P}_{i,j}} \min_{k<|\mathsf{p}|} \hat{p}\left(\frac{1}{2}(x_{\mathsf{p}_k} + x_{\mathsf{p}_{k+1}})\right) \\ &\quad \approx \quad c \cdot \exp\left[-\frac{1}{2\sigma^2}\left(\min_{\mathsf{p} \in \mathsf{P}_{i,j}} \max_{k<|\mathsf{p}|} d(\mathsf{p}_k, \mathsf{p}_{k+1})\right)^2\right] \quad (1) \\ &\quad \equiv \quad k(\mathbf{x}_i, \mathbf{x}_j). \end{aligned}$$

This k, called "connectivity kernel", is positive definite and was suggested for clustering previously [10].

The kernel values do not depend on the length of the paths, which may lead to the connection of otherwise separated clusters by single outliers ("bridge" points). To avoid this problem, we "soften" the max in Equation (1) by replacing it with

$$\mathrm{smax}^\rho(\mathsf{p}) := \frac{1}{\rho} \ln\left(1 + \sum_{k=1}^{|\mathsf{p}|-1} e^{\rho d(\mathsf{p}_k, \mathsf{p}_{k+1})} - 1\right). \quad (2)$$

Equation (1) is recovered by taking $\rho \to \infty$. If $\rho \to 0$, $\mathrm{smax}^\rho(\mathsf{p})$ becomes simply the sum of original distances along the path $\mathsf{p} \in \mathsf{P}_{i,j}$. Due to the triangular inequality, this is never less than $d(i, j)$, so that in a full graph with Euclidean distances the minimum path distance

[1] made symmetric by including (j, i) in E if $(i, j) \in E$

becomes $||\mathbf{x}_i - \mathbf{x}_j||_2$. Thus, the standard Gaussian RBF kernel is recovered, and no use is made of the unlabeled data. However, for a sparse graph computing the minimum path distance when $\rho \to 0$ is equivalent to `Isomap` [19].

The proposed method can be summarized as follows:

1. Build nearest neighbor graph G from all (labeled and unlabeled) data.
2. Compute the $n \times (n+m)$ distance matrix D^ρ of minimal ρ-path distances according to
$$D^\rho_{i,j} = \frac{1}{\rho^2} \ln \left(1 + \min_{\mathsf{p} \in \mathsf{P}_{i,j}} \sum_{k=1}^{|\mathsf{p}|-1} e^{\rho d(\mathsf{p}_k, \mathsf{p}_{k+1})} - 1 \right)^2$$
from all labeled points to all points.
3. Perform a non-linear transformation on D^ρ to get kernel K,
$$K_{i,j} = \exp\left(-\frac{D^\rho_{i,j}}{2\sigma^2} \right)$$
The linear case corresponds to $\sigma = \infty$ and $K = -\frac{1}{2} H^n D^\rho H^{n+m}$, with H^p being the $p \times p$ centering matrix (as in Multidimensional Scaling [8]): $H^p_{ij} = 1_{i=j} - 1_{i \le n}/n$.
4. Train an SVM with K and predict.

2.1.3 Comments

A few comments can be made on these steps.

1- The use of a sparse graph G is merely a way to save computation time. This is in contrast to some other graph-based methods, that require sparseness for detecting the manifold structure (e.g. `Isomap`). In our method, the sparse graph is always seen as an approximation to the full graph. However, the accuracy of this approximation depends on the value of the softening parameter ρ: for $\rho \to 0$, the direct connection is always shortest, so that every deletion of an edge can cause the corresponding distance to increase. For $\rho \to \infty$, shortest paths almost never contain any long edge, so that long edges can safely be deleted.

2- For large values of ρ, the distances between points in the same cluster are decreased. In contrast, the distances between points from different clusters are still dominated by the gaps between the clusters and, as a result, those gaps become more pronounced.

Instead of Equation (2), it is possible to use other interpolations between the max and the mean such as the Minkowski metric, $\left(\sum_{k=1}^{|\mathsf{p}|-1} d(\mathsf{p}_k, \mathsf{p}_{k+1})^{\rho+1} \right)^{1/(\rho+1)}$.

3- K is in general not positive definite (p.d.), except for $\rho = 0$ (standard RBF) and $\rho = \infty$ (then D^ρ is an ultrametric and thus negative definite [10], yielding a p.d. kernel [17]). In practice, negative eigenvalues can be observed, but they are few and small in absolute value, as documented in Table 1. In our experiments, the SVM training still converges quickly. Moreover, recent papers have argued in favor of the use of non-positive definite kernels for learning [11, 16].

ρ	0	0.5	1	2	4	8	∞
ν	0	0.19	2.96	4.66	1.89	0.02	0

Table 1: Empirically found weight on the `Coil20` dataset of the negative eigenvalues as percentage of the weight of all eigenvalues, $\nu := 100 \sum_i \max(0, -\lambda_i) / \sum_i |\lambda_i|$.

2.2 MARGIN MAXIMIZATION

The Transductive Support Vector Machine (TSVM), first introduced in [20] and implemented by [3, 12], aims at minimizing the following functional,

$$\min \frac{1}{2}\mathbf{w}^2 + C \sum_{i=1}^{n} \xi_i + C^* \sum_{i=n+1}^{n+m} \xi_i,$$

under the constraints:

$$y_i \quad \begin{array}{ll} (\mathbf{w} \cdot \mathbf{x}_i + b) \ge 1 - \xi_i & 1 \le i \le n \\ |\mathbf{w} \cdot \mathbf{x}_i + b| \ge 1 - \xi_i & n+1 \le i \le n+m \end{array} .$$

This can be rewritten without constraint as the minimization of

$$\frac{1}{2}\mathbf{w}^2 + C \sum_{i=1}^{n} L(y_i(\mathbf{w} \cdot \mathbf{x}_i + b)) + C^* \sum_{i=n+1}^{n+m} L(|\mathbf{w} \cdot \mathbf{x}_i + b|), \tag{3}$$

with $L(t) = \max(0, 1-t)$.

Unfortunately, the last term makes this problem non-convex and difficult to solve [3, 12]. The implementation of TSVM that we propose in this paper is to perform a standard gradient descent on (3). However, since this latter is not differentiable, we replace it by

$$\frac{1}{2}\mathbf{w}^2 + C \sum_{i=1}^{n} L^2(y_i(\mathbf{w} \cdot \mathbf{x}_i + b)) + C^* \sum_{i=n+1}^{n+m} L^*(\mathbf{w} \cdot \mathbf{x}_i + b), \tag{4}$$

with $L^*(t) = \exp(-3t^2)$ (c.f. Figure 2).

To enforce that all unlabeled data are not put in the same class, we add the additional constraint,

$$\frac{1}{m} \sum_{i=n+1}^{n+m} \mathbf{w} \cdot \mathbf{x}_i + b = \frac{1}{n} \sum_{i=1}^{n} y_i. \tag{5}$$

This is in analogy to the treatment of the min-cut problem in spectral clustering, which is usually replaced by the normalized cut to enforce balanced solutions [13].

Finally, note that unlike traditional SVM learning algorithms, which solve the problem in the dual, we directly solve the problem in the primal. If we want to use a non-linear kernel, it is possible to compute the coordinates of each point in the kernel PCA basis [17]. More directly, one can compute the Cholesky decomposition of the Gram matrix, $K = \tilde{X}\tilde{X}^\top$ and minimize (4) with $\mathbf{x}_i \equiv (\tilde{X}_{i,1} \dots \tilde{X}_{i,n+m})$.

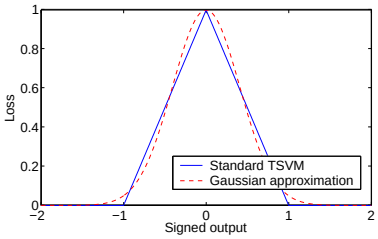


Figure 2: TSVM cost functions for unlabeled data.

We decided to initially set C^* to a small value and increase it exponentionally to C; thereby following `SVMlight`. Note that the choice of setting the final value of C^* to C is somewhat arbitrary. Ideally, it would be preferable to consider this value as a free parameter of the algorithm.

2.3 IMPLEMENTATION

From the methods discussed above, we derive three algorithms:

1. `graph`, training an SVM on a graph-distance derived kernel;
2. `∇TSVM`, training a TSVM by gradient descent;
3. `LDS` (Low Density Separation), combining both of the previous algorithms.

For `SVM`, we use the `Spider`[2] machine learning package for `matlab`. For `∇TSVM`, a conjugate gradient descent method was used.[3]

The distance computation for `graph` can be carried out using the shortest path algorithm by Dijkstra [9]. For `LDS`, the full $(n+m) \times (n+m)$ matrix D^ρ of pairwise distances has to be computed.

[2]available at `http://www.kyb.tuebingen.mpg.de/bs/people/spider`

[3]available at `http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/minimize`

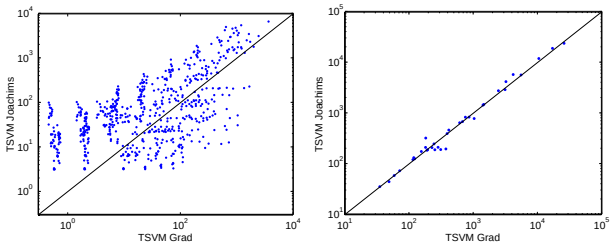


Figure 3: Each point represents the values of the objective function reached by the `TSVM` and `∇TSVM` for some value of C, σ. Points above the diagonal mean that `∇TSVM` found a better local minimum. Left: `Coil20` dataset, right: `g10n` (both described below).

Since the derived kernel is (in general) not positive definite, we can apply Multidimensional Scaling (MDS) [8] to find a Euclidean embedding of D^ρ before applying `∇TSVM`. The embedding found by the classical MDS are the eigenvectors corresponding to the positive eigenvalues of $-HD^\rho H$, where $H_{ij} = \delta_{ij} - 1/(n+m)$. For computational reasons, we decided to take only the first p eigenvectors such that

$$\sum_{i=1}^{p} \lambda_i \geq (1-\delta) \sum \max(0, \lambda_i) \quad \text{and} \quad \lambda_p \leq \delta\lambda_1, \quad (6)$$

with decreasing eigenvalues $\lambda_1 \geq \dots \geq \lambda_{n+m}$.

We compare our algorithms to one state of the art supervised method, `SVM`, and to two state of the art semi-supervised methods, the `TSVM` optimization scheme as implemented in `SVMlight` [12] and a graph-based `manifold` learning, which is closely related to those in [1, 22, 23]. More precisely, we estimate the labels of the unlabeled points by minimizing the functional

$$\sum_{i=1}^{n}(f_i - y_i)^2 + \frac{\lambda}{\sum_{i,j} w_{ij}} \sum_{i,j=1}^{n+m} (f_i - f_j)^2 w_{ij}, \quad (7)$$

where $w_{ij} = \exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2)$ if $\mathbf{x}_i$ is among the k nearest neighbors of $\mathbf{x}_j$ (or vice-versa), and 0 otherwise. This methods depends on the sparsity of the graph.

Figure 3 compares how both implementations of TSVM are able to minimize the cost function (3). Note that our proposed implementation does not minimize (3), but the differentiable approximation (4) and for this reason it has a disadvantage in the comparison shown in Figure 3. Nevertheless, on average it produces better values of the objective function, which translate, as we will see later, into better test errors.

2.3.1 Computational Complexity

We implement the search for the next-closest unexplored node in Dijkstra's algorithm with a prior-

ity queue based on a binary heap. This results in $\mathcal{O}(|E| \log(n+m))$ run time for computing the path distances of one labeled point to all other points. Thus, the entire matrix D^ρ costs $\mathcal{O}(nk(n+m)\log(n+m))$ on a k-NN graph.

The time complexity of a gradient descent algorithm is approximately equal to that of evaluating the cost function multiplied by the square of the number of variables. For ∇TSVM , this amounts to $\mathcal{O}\ (n+m)^3$. The MDS is of the same time complexity, since it computes the eigendecomposition of an $(n+m)\times(n+m)$ matrix. For both algorithms, the complexity can be reduced if one considers only the first p eigenvectors.

While ∇TSVM needs to store the entire kernel matrix (on both labeled and unlabeled points), for graph an $n \times (n+m)$ part is sufficient. Memory can be reduced to the $n \times n$ part required for SVM training, but the (worst case) time required to compute individual shortest paths is as much as is required for computing all paths from a single source to all targets. For both SVM and TSVM, in practice only parts of the kernel matrices have to be (computed and) stored, because of the sparsity of the solution.

For training the manifold algorithm as given in Eq. 7, a sparse $(n+m)\times(n+m)$ matrix needs to be stored and inverted. Due to the use of a k-NN graph, the matrix has about $k(n+m)$ entries (at most $2k(n+m)$).

2.3.2 Parameters

For each algorithm, the values for a number of parameters have to be fixed. In practical applications, this is usually done by cross-validation (CV). While this is no major problem for two parameters (like the SVMs have), it is impractical for the five parameters of the graph algorithm. To reduce this number, we fix three of them in advance, as shown in the table:

algorithm	free parameters; [fixed parameters]
SVM	σ, C
TSVM	σ, C
manifold	σ, k, λ
∇TSVM	σ, C
graph	C, ρ; [$\sigma=\infty$, $k=n+m$, $\delta=0.1$]
LDS	C, ρ; [$\sigma=\infty$, $k=n+m$, $\delta=0.1$]

Figure 4 demonstrates that for LDS the parameter fixing proposed above leads only to a minor loss in accuracy. As shown in (a), a fully connected graph is good (for the optimum value of ρ). As shown in (b), $\sigma=\infty$ (i.e. no further non-linear transformation) is good (again, for the optimum value of ρ). In general the resulting kernel will not be positive definite (except for $\rho=0$ and $\rho=\infty$, see also Table 1). As shown in (c), the SVM seems to handle negative eigenvalues reasonably well. This can be seen on the right of (c): almost the same results were obtained with and without MDS. It seems safe to discard the eigenvectors corresponding to small (positive) eigenvalues (c.f. left side of the plot (c)). In the rest of the experiments, we set $\delta=0.1$.

To determine good values of the remaining free parameters (eg, by CV), it is important to search on the right scale. We therefore fix default values for C and σ that have the right order of magnitude. In a c-class problem, we use the $1/c$ quantile of the pairwise distances $D^\rho_{i,j}$ of all data points as the default for σ. The default for C is the inverse of the empirical variance s^2 of the data in feature space, which can be calculated by $s^2 = \frac{1}{n}\sum_i K_{ii} - \frac{1}{n^2}\sum_{ij} K_{ij}$ from a $n\times n$ kernel matrix K. Below, all values for these parameters will be given relative to the respective default values, making them comparable for different data sets.

2.3.3 LDS algorithm

The final LDS algorithm is summarized in Figure 1. Note that slight changes are required for the extreme settings of ρ: for $\rho=0$, steps 1 to 3 have to be replaced by simply running the shortest path algorithm on $d(i,j)$ to compute $d_{i,j}$; for $\rho=\infty$, a modified version of Dijkstra that keeps track of maximum distances instead of sums along paths must be used. A matlab implementation of LDS can be obtained at http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/lds/.

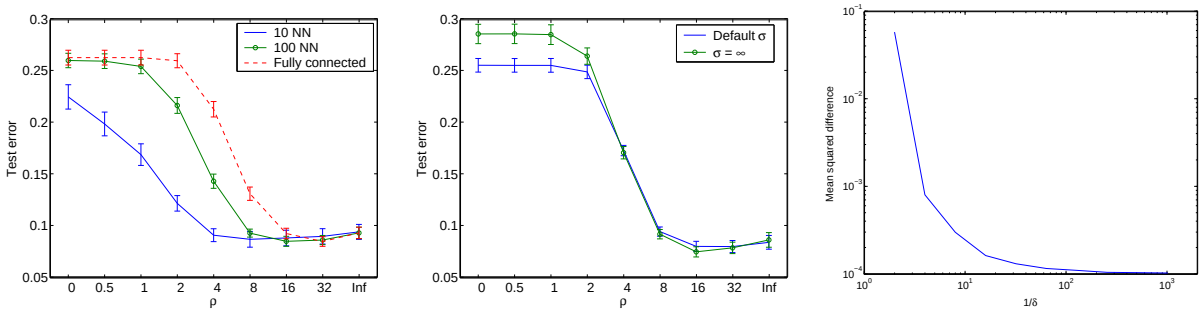
3 EXPERIMENTAL RESULTS

3.1 DATA SETS

In order to get a good picture of the effectiveness of the algorithms, we compare their generalization performance on two artificial and three real world data sets with different properties.

data set	classes	dims	points	labeled
g50c	2	50	550	50
g10n	2	10	550	50
Coil20	20	1024	1440	40
Text	2	7511	1946	50
Uspst	10	256	2007	50

The artificial data sets are inspired by [2]: the data are generated from two standard normal multi-variate Gaussians. In g50c, the labels correspond to the Gaussians, and the means are located in 50-dimensional space such that the Bayes error is 5%. In contrast, g10n is a deterministic problem in 10 dimensions, where the decision function traverses the centers of the Gaussians (thus violating the cluster assumption), and depends on only two of the input dimensions.



(a) (b) (c)

Figure 4: Influence of parameter choice on the test error of LDS on the `Coil20` data: (a) the graph structure; (b) σ; and (c) the approximation accuracy of the MDS. Plot (c) shows the square difference between the test error achieved with and without MDS, averaged over different values of C and ρ.

Algorithm 1 `LDS` algorithm

Require: ρ, C

 # Compute ρ-distances:

Build a fully connected graph with edge lengths $w_{ij} = \exp(\rho d(i,j)) - 1$.

Use Dijkstra's algorithm [9] to compute the shortest path lengths $d_{SP}(i,j)$ for all pairs of points.

Form the matrix D of squared ρ-path distances by $D_{ij} = \left(\frac{1}{\rho}\log(1 + d_{SP}(i,j))\right)^2$.

 # Perform multidimensional scaling:

$U\Lambda U^\top = -HDH$, where $H_{ij} = \delta_{ij} - 1/(n+m)$.

Find the threshold p such that (6) holds.

The new representation of $\mathbf{x}_i$ is $\tilde{x}_{ik} = U_{ik}\sqrt{\lambda_k}, 1 \le k \le p$.

 # Train TSVM:

for i=0 to 10 **do**

  Set $C^* = 2^{i-10}C$

  Minimize by gradient descent (3) under constraint (5).

end for

The real world data sets consist of two-class and multi-class problems. In `Coil20`, the data are gray-scale images of 20 different objects taken from different angles, in steps of 5 degrees [15]. The `Text` dataset are the classes `mac` and `mswindows` of the `Newsgroup20` dataset preprocessed as in [18]. Finally, our `Uspst` set contains the test data part of the well-known `USPS` data on handwritten digit recognition.

3.2 EXPERIMENTS

For each of the data sets, 10 different splits into labeled and unlabeled points were randomly generated. We took care to include at least one point of each class in the labeled set (two for `Coil20`).

We used a different model selection strategy for `LDS` than for the other algorithms. For `LDS`, we carry out 5-fold cross-validation (CV) on the training set for each split, thereby simulating the real world application scenario. Note that all data (training and test) can be (and is) used as unlabeled data. The reported test errors are obtained after training the selected model on the entire training set. For the other algorithms, we are interested in the best possible performance, and simply select the parameter values minimizing the test error. In both cases, we select combinations of values on a finite grid as follows:

parameter	values
width σ	$2^{-3}, 2^{-2}, 2^{-1}, 2^0, 2^1, 2^2, 2^3$
exponent ρ	$0, 2^0, 2^1, 2^2, 2^3, 2^4, +\infty$
penalty C	$10^{-1}, 10^0, 10^1, 10^2$
degree k	$10, 100$, all
regulariz. λ	$4^{-2}, 4^{-1}, 4^0, 4^1, 4^2$

Although `LDS` and `graph` work with any kernel, we here fix the linear kernel ($\sigma = \infty$; c.f. section 2.3.2).

3.3 RESULTS

The results are presented in Table 2. Except for the data set `g10n`, `LDS` always achieves lower test errors with empirically found parameter settings than all the

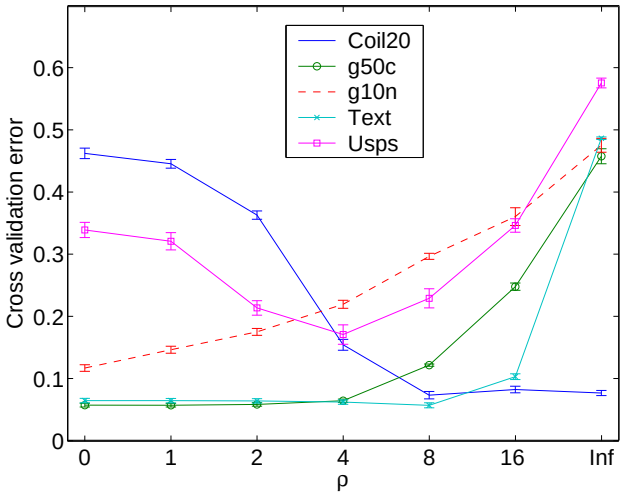


Figure 5: Cross-validation error (with standard deviation error bars) as a function of the parameter ρ.

other algorithms are capable of achieving, even when optimal parameter settings are known. This clearly demonstrates the superiority of `LDS`.

Although ∇`TSVM` always performs better (and usually, significantly better) than `TSVM`, it still fails to reach the level of `manifold` on the `Coil20` data set. But this shortcoming is eliminated by making use of the `graph` transform of the distances.

To better understand the role of the distance transform, we depict the 5-fold cross validation error for the best value of C, averaged over the 10 splits, as a function of ρ in Figure 5. We can distinguish three cases: the minimum is (i) at or close to 0; (ii) at or close to ∞; or (iii) somewhere in between.

(i) Linear classifiers are optimal by construction for the artificial data, and likely to be optimal for `Text` due to the high dimensionality. For `g50c` and `Text`, $\rho > 0$ does not substantially help ∇`TSVM`, but does not hurt either. Only for `g10n`, where the cluster assumption does not hold, increasing ρ immediatly increases the test error. **(ii)** In `Coil20`, the points of each class lie eqi-distantly on a ring. With $\rho = \infty$, all their pairwise distances are reduced to the distance of two neighboring points. Note that there is no noise which could cause unwanted bridging between two classes. **(iii)** Perhaps the most interesting case is `Uspst`, with an optimum of $\rho = 4$. While there definitely are clusters corresponding to the classes, there seem to exist outliers that would, for too large ρ, lead to erroneous merging of clusters.

As the optimum value of ρ seems to correspond to features of the data set, prior knowledge on the data could possibly be used to narrow the range to be searched.

4 CONCLUSIONS

The TSVM objective function could, at a first sight, be interpreted as a straight-forward extension of the maximum margin principle of SVM to unlabeled data. We conjecture that it actually implements two different principles: the regularization by margin maximization on the labeled points, and the cluster assumption by margin maximization on the unlabeled points. The latter does not lead to smoother decision functions, but it enforces that the decision boundary lies in low density regions.

The strength of our gradient descent approach might be that it directly optimizes the objective according to the cluster assumption: to find a decision boundary that avoids high density regions. In contrast, `TSVM` (`SVMlight` implementation) might suffer from the combinatorial nature of its approach. By deciding, from the very first step, on the putative label of every point (even though with low confidence), it may lose important degrees of freedom at an early stage, and get trapped in a bad local minimum.

The pairwise distances computed by the `graph` algorithm attempt to reflect the cluster assumption: distances of points from the same cluster are shrunk, while for points in different clusters they are dominated by the inter-cluster distance. Used with an SVM, this clearly improves over standard (Euclidean) distances, but not over other semi-supervised methods.

The combination of the `graph` distance computation with the TSVM training yields a clearly superior semi-supervised algorithm. Apparently the preprocessed distances make it less likely for the TSVM to get stuck in very suboptimal local minima. Probably the preprocessing widens small density valleys so that they are more readily found by local searches.

Although manifold learning indirectly exploits the cluster assumption, as argued above, another feature may contribute to its successes. If the intrinsic dimensionality of the data manifolds is much smaller than that of the input space, restricting the learning process to the manifolds can alleviate the "curse of dimensionality". We plan to investigate how much performance can be gained in this manner.

Future work will be on a thorough comparison of discriminative semi-supervised learning methods. We observe that the time (and to some degree, also space) complexities of all methods investigated here prohibit the application to really large sets of unlabeled data, say, more than a few thousand. Thus, work should also be devoted to improvements of the computational efficiency of algorithms, ideally of `LDS`.

	methods from literature			proposed methods		
data set	SVM	manifold	TSVM	graph	∇TSVM	LDS
Coil20	24.64%	6.20%	26.26%	6.43%	17.56%	**4.86%**
g50c	8.32%	17.30%	6.87%	8.32%	5.80%	**5.62%**
g10n	9.36%	30.64%	14.36%	9.36%	9.82%	9.72%
Text	18.87%	11.71%	7.44%	10.48%	5.71%	**5.13%**
Uspst	23.18%	21.30%	26.46%	16.92%	17.61%	15.79%

Table 2: Mean test error rates. Note that model selection was done by cross-validation for LDS whereas by minimizing the test error for the other methods. Bold numbers are statistically significantly (95% confidence) better compared to all other methods.

Acknowledgements

We thank Bernhard Schölkopf and Matthias Hein for valuable comments.

References

[1] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In *COLT*, 2004.

[2] Y. Bengio and Y. Grandvalet. Semi-supervised learning by entropy minimization. In *NIPS*, volume 17, 2004.

[3] K. Bennett and A. Demiriz. Semi-supervised support vector machines. In *NIPS*, volume 12, 1998.

[4] O. Bousquet, O. Chapelle, and M. Hein. Measure based regularization. In *NIPS*, 2004.

[5] O. Chapelle. *Support Vector Machines: Induction Principle, Adaptive Tuning and Prior Knwoledge*. PhD thesis, LIP 6, 2003.

[6] O. Chapelle, J. Weston, L. Bottou, and V. Vapnik. Vicinal risk minimization. In *NIPS*, volume 13, 2000.

[7] O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In *NIPS*, volume 15, 2002.

[8] T. F. Cox and M. A. Cox. *Multidimensional Scaling*. Chapman & Hall, 1994.

[9] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Math.*, 1:269–271, 1959.

[10] B. Fischer, V. Roth, and J. M. Buhmann. Clustering with the connectivity kernel. In *NIPS*, volume 16, 2004.

[11] B. Haasdonk. Feature space interpretation of SVMs with indefinite kernels. *IEEE TPAMI*, 2004. In press.

[12] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, pages 200–209, 1999.

[13] T. Joachims. Transductive learning via spectral graph partitioning. In *ICML*, 2003.

[14] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *ICML*, 2002.

[15] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (coil-20). Technical Report CUCS-005-96, Columbia Univ., USA, February 1996.

[16] C. S. Ong, X. Mary, S. Canu, and A. J. Smola. Learning with non-positive kernels. In *ICML*, pages 639–646, 2004.

[17] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[18] M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *NIPS*, volume 14, 2001.

[19] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[20] V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

[21] P. Vincent and Y. Bengio. Density-sensitive metrics and kernels. Presented at the Snowbird Learning Workshop, 2003.

[22] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, volume 16, 2003.

[23] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003.

Learning spectral graph segmentation

Timothée Cour
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

Nicolas Gogin
Computer Science
Ecole Polytechnique
91128 Palaiseau Cedex, FRANCE

Jianbo Shi
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

Abstract

We present a general graph learning algorithm for spectral graph partitioning, that allows direct supervised learning of graph structures using hand labeled training examples. The learning algorithm is based on gradient descent in the space of all feasible graph weights. Computation of the gradient involves finding the derivatives of eigenvectors with respect to the graph weight matrix. We show the derivatives of eigenvectors exist and can be computed in an exact analytical form using the theory of implicit functions. Furthermore, we show for a simple case, the gradient converges exponentially fast. In the image segmentation domain, we demonstrate how to encode top-down high level object prior in a bottom-up shape detection process.

1 INTRODUCTION

Image segmentation and data clustering are two fundamental operations in computer vision and machine learning. Let $\mathbf{I} = \{\mathbf{x_1}, ..., \mathbf{x_n}\}$ be a set of feature vectors representing n image pixels or data points. The image segmentation process partitions pixels into K disjoint groups. In a 2-way segmentation, we seek an output vector $SEG(\mathbf{I}) = \{y_1, ..., y_n\} \in \{0,1\}^n$, such that a segmentation goodness measure is optimized. We defined segmentation as a mapping from $\mathbf{x_i}$ to $y_i \in \{0,1\}$ to purposely hint its potential connection to a supervised learning method we should propose. Our goal is to *teach* the image segmentation through a set of hand labeled training examples. Given a set of image/segmentation pairs $\{\mathbf{I^i}, SEG^*(\mathbf{I^i})\}$, the system will learn to adjust so that the computed segmentation $SEG(\mathbf{I^i})$ is close to $SEG^*(\mathbf{I^i})$. With a *supervised* image segmentation, we are able to encode top-down object familiarity prior in a bottom-up distributed process. In this paper, we will demonstrate a system that can detect and segment rectangular shaped objects in a clutter image background by learning from examples.

To appreciate why learning image segmentation is difficult, we summarize below its basic principles. Segmentation algorithms are defined by the clustering criteria and computational process to optimize it. For example, in the Markov Random Field (MRF) formulation, the criteria is to maximize $P(SEG(\mathbf{x})|\mathbf{x}) = \frac{1}{Z} exp(\sum -f(y_i, y_j|\mathbf{x}))$, where $f(y_i, y_j|\mathbf{x})$, called clique potential, specifies a local measure of grouping pixel i with j. While each $f(y_i, y_j|\mathbf{x})$ can be easily corrupted, the global optimum of $P(SEG(\mathbf{x})|\mathbf{x})$ must balance preferences on all pairs of $f(y_i, y_j|\mathbf{x})$ and therefore is stable. Spectral graph partitioning, such as Normalized Cut (Ncut)[6][5], has been developed as a computationally efficient alternative to MRF. Image segmentation is mapped to a graph partitioning problem, where the graph consists of the pixels/data points as nodes, and the weighted graph edges $W(i,j)$ serve as the equivalent of Clique Potential $f(y_i, y_j|\mathbf{x})$. The global segmentation criterion Ncut seeks a balanced segmentation and grouping of the pixels. Computationally the solution is derived from the eigenvectors of $Wy = \lambda Dy$, where D is the degree matrix. As in the MRF case, the eigenvectors are implicitly related to the input weight matrix W, and are quite insensitive to random perturbation of W.

While global decision process from local feature comparison brings a stable segmentation, it makes the learning segmentation a difficult task. Treating any segmentation learning algorithm as a black box, one must be able to back-trace error on the output of global segmentation to the input local clique potential or pair-wise weight matrix. Since the global decision is only implicitly related to the input, it is hard to explicitly *assign* a blame to a particular clique potential or weight matrix entry. To account for segmentation error on just *one pixel*, we would potentially need to adjust *all* possible pairs of clique potential or weight

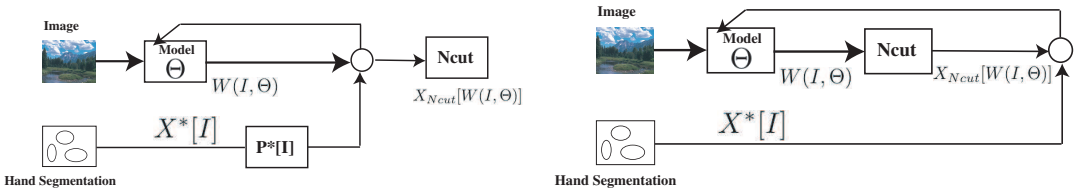


Figure 1: Two alternative algorithms for learning spectral graph partitioning. (A) methods of Meila-Shi[4] optimizes the graph weight $W(I,\Theta)$ by minimizing the KL-divergence between an equivalent random walk matrix $P(I,\Theta)$ and the target $P^*(I,\Theta)$. (B) Our method directly optimize the error on the output Ncut segmentation vector $X_{Ncut}[W(I,\Theta)]$ by gradient descent in the space of all feasible graph weights using explicit computation of the derivatives of eigenvectors.

matrix entries!

Meila-Shi[4] first studied the problem of learning spectral graph cuts with supervised training data. Their proposed algorithm learned the graph weight W_{ij} by minimizing the KL-divergence between an equivalent random walk matrix P_{ij} and the target P^*_{ij} derived from the hand labeled segmentation. However the formulation provides no explicit constraints on the Ncut eigenvector itself. Bach-Jordan[1] formulated a direct optimization of W with respect to its Ncut eigenvector. They transform the *implicit* relationship between W and Ncut eigenvector into an *explicit* one by making a differentiable approximation of eigenvector using power method. The resulting computation of derivatives of eigenvector is however complex and can be computationally unstable.

We present in this paper a direct method for learning spectral graph cut, based on efficient computation of derivatives of Ncut eigenvectors in *exact analytical form*. We show that there is an *explicit* computation that assigns the segmentation error to the input graph weight matrix. This capability allows us to design parameterized graphs that can encode and detect complex objects. The paper is organized as follows. We describe in Sec. 2 the structure of the graph we use for image and shape segmentation. Sec. 3 describes the learning algorithm and its convergence properties. We show our results in Sec. 4.

2 PROBLEM SETUP

We will demonstrate a learnable segmentation algorithm for detecting and segmenting desired object shape such as a rectangle in an image. The shape detection-segmentation process begins with edge detection. Each edge i is parametrized by (x_i, y_i, θ_i), its location and orientation. Denote $F(I) = \{e_1, ..., e_k | e_i = (x_i, y_i, \theta_i)\}$ the set of edges detected for image I, and F the complete set of possible edges detected in all images. The goal of the segmentation algorithm is to group the edges which form a rectangle, and separate them from background edge clutters, as shown in Fig. 11.

While a rectangle is a relatively simple shape, its aspect can be quite flexible with variable aspect ratio in x, y, and variable orientation. Assuming we have quantized the orientation into N_{angle} angles, for an image size of $N_{pixel} \times N_{pixel}$ a brute force method would need to search over $O(N^4_{pixel}N_{angle}$ possible configurations (for a 100×100 image with 10 orientations quantization, we have 1 billion configurations!). One way to avoid this large scale search is to decompose the rectangles into simple local configurations (corners, lines, parallel lines), and combine them by checking their global consistency. This data-driven bottom-up process only needs to check roughly $O(N_{edge}) = O(|F(I)|)$ local configurations (assuming a fixed neighborhood size). The global integration can be carried out in the grouping framework of graph partitioning such as Ncut, which has empirically a running time of $O(N^{1.5}_{edge})$. Furthermore, the decomposition of a shape into local edge relationships also makes the detection more robust to image background clutter.

2.1 LOCAL SHAPE CONFIGURATIONS

We need to define functions on local configuration goodness, with the hope of discriminating rectangular object vs. background. Since we are using oriented edges, we can favor convex configurations and penalize concave or other impossible configurations, as illustrated in Fig. 2. The function that assesses the goodness of a particular configuration is denoted as clique potential: $f(e_1, \ldots, e_K)$ is high only when $(e_1, \ldots, e_K)$ form a familiar configuration. The problem of designing this clique potential can be quite complex in general. For example, consider the case of binary relationships: we need to find a potential function for all possible pairs of edges (e_1, e_2):

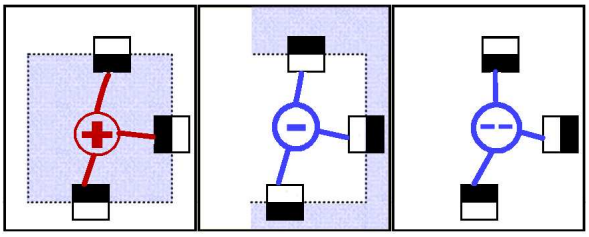


Figure 2: Different oriented edge configurations and associated clique potential. Left: three edges forming a convex object (likely to be found in rectangle shapes). Middle: concave configuration (unlikely in rectangular shapes). Right: impossible configuration (very unlikely to be found in any object).

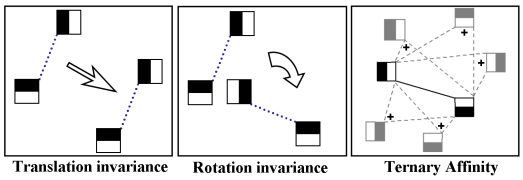


Figure 3: Properties of the clique potential/affinity matrix. Left and middle: translation and rotation invariance, $f(x_1, y_1, \theta_1; x_2, y_2, \theta_2) = \bar{f}(x_2 - x_1, y_2 - y_1, \theta_1, \theta_2)$ and, $f(z_1, \theta_1; z_2, \theta_2) = \bar{\bar{f}}((z_2 - z_1)e^{-i\theta_1}, \theta_2 - \theta_1)$. Right: summing up ternary affinities to obtain a binary affinity, $f(e_1, e_2) = \sum_i f_3(e_1, e_2, e_i)$.

$f(e_1, e_2) = f(x_1, y_1, \theta_1, x_2, y_2, \theta_2)$. The function takes 4-dimensional inputs, and even in the simple case of 10x10 possible edge locations, with 4 orientations $\{\pi/2, 2\pi/2, 3\pi/2, 4\pi/2\}$, that makes 160,000 different values to design through learning. To make the learning problem more managable, we use the following parameterization that induces translational invariance:

$$f(x_1, y_1, \theta_1; x_2, y_2, \theta_2) = \bar{f}(x_2 - x_1, y_2 - y_1, \theta_1, \theta_2) \quad (1)$$

If in addition we also require invariance by rotation, the use of complex numbers comes in handy, with $z = x + iy$ we obtain $f(z_1, \theta_1; z_2, \theta_2) = \bar{\bar{f}}((z_2 - z_1)e^{-i\theta_1}, \theta_2 - \theta_1)$. These invariance properties are illustrated in Fig. 3.

2.2 GLOBAL SHAPE DETECTION FROM LOCAL CONFIGURATIONS

With local edge clique function we could eliminate wrong patterns of edges, retrieve the correct edge orientation when ambiguous, and enhance good configurations. However, there are many ambiguous cases in which local properties are insufficient to decide the foreground/background labeling. Think about a weak edge at object boundary, or a strong clutter edge in the background. A direct thresholding technique would fail here. Another example is provided by Fig. 4, where a local approach would favor the wrong edge orientation. As in the case of image segmentation, local grouping measures need to be aggregated to form a global segmentation decision. We will see in the next section how to formulate this precisely in graph framework, through Spectral Graph Partitioning.

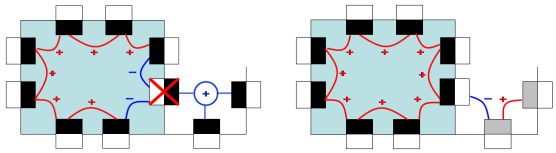


Figure 4: Each edge has 2 hypothesized opposite polarities. We want to inhibit clutter edges and recover correct polarity. Left: local segmentation of edges produces the wrong polarity for one edge (barred), grouping it with clutter. Right: global aggregation of edge affinities yields a correct grouping and inhibits clutter.

2.3 SPECTRAL GRAPH PARTITIONING FORMULATION

Such local relationships between image features are well captured by the notion of graph $G = \langle V, W \rangle$. The graph nodes V consist of the image edge features $F = \{e_i\}$, and the graph edges are the relationships between the edge features with affinity matrix $W \in \mathbb{R}^{n \times n}$ defined by $W_{ij} = f(e_i, e_j)$. Higher-order edge feature relationships can be translated into binary affinities by summing over cliques: $W_{ij} = \sum_{i_1=i, i_2=j, i_3, \ldots, i_K} f(e_{i_1}, \ldots, e_{i_K})$, as illustrated in Fig. 3. We denote $V(I)$ the image edge features, $F(I)$, detected in I; $W(I) = W(V(I), V(I))$, the subgraph affinity induced by image features in I.

Let us recall our goal: we want to partition the graph nodes $V(I)$ into two groups, using an indicator vector X: $X_i = 1$ if detected feature $V(I)_i$ belongs to foreground, and $X_i = -1$ if it belongs to background. The segmentation process should ensure that edge features (nodes) grouped together have high mutual *affinity*, and nodes in different sets have low *affinity*. We will use the Normalized Cuts (Ncut) criterion for the segmentation process. Ncut criterion can be optimized by finding the second generalized eigenvector of $(W(I), D(I))$ ($D(I)$ is the degree matrix of $W(I)$):

$$W(I)X(I) = \lambda_2 D(I)X(I) \quad (2)$$

$X(I)$ is then thresholded to determine the foreground/background labelling. Note that, the solution we obtain for $X(I)$ is an implicit function of the weight matrix $W(I)$, which is defined by the local

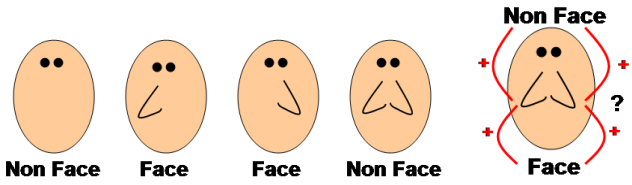


Figure 5: The XOR function. Suppose we have a face detection graph with nodes: Left Nose (LN), Right Nose (RN), Face (F), and Non-Face(NF). The Hebbian learning rule, based on feature coocurrence, would find the following weights: $W(LN,F) = W(RN,F) = W(LN,NF) = W(RN,NF) = \frac{1}{2}$, making it impossible to distinguish between a Face and a Non-Face. The graph learning algorithm we propose does not suffer from this.

clique potentials, $f(e_i, e_j)$. However, its computation is tractable and, as we shall see, we can apply perturbation theory to analyze their effect on the segmentation task. This last point is essential: it means that we can *assign a blame* to the local graph structure, by looking at the global segmentation result. Hence the system is not a black box anymore, we can train it.

3 LEARNING THE GRAPH STRUCTURE

As we have seen, the design of the clique potential is as crucial to the segmentation as it complex. In the Ncut formulation, whether we use a parameterization of the affinity matrix $W(\Theta)$ or a direct representation through its coefficients W_{ij}, real image segmentation tasks will require a large number of parameters to be optimized. Hence the need for a principled algorithm to learn the clique potential.

3.1 WHY ARE SIMPLE LEARNING SCHEMES INSUFFICIENT

One natural idea in learning the graph clique potential is simply to measure the coocurrence of image features accross a set of training images, in accordance to the Hebbian rule. This rule strengthens the weight W_{ij} if feature i and feature j are strongly correlated, according to: $W_{ij} = \sum_I V(I)_i V(I)_j$ in our notation. Though intuitive, this rule is insufficient for our problem. Fig. 5 illustrates a typical situation that the Hebbian rule is unable to handle, namely the XOR boolean function. More generally, the Hebbian rule cannot learn non-linearly separable functions. We have shown in [2][1] that our system does not have this limitation and it could learn XOR.

[1]http://www.seas.upenn.edu/∼timothee/research.html

3.2 PRINCIPLE OF LEARNING

The Maximum Likelihood formulation (ML) tries to adjust the clique potential so that it maximally explains the data (the set of training images). However, this formulation doesn't take into account the graph inference procedure $I \rightarrow X(I)$, as a result, it can produce a probability distribution that cannot be inferenced efficiently. We use a different approach. We adjust the clique potential so that the output of the system gets closer to the desired segmentation. In the following, we assume we are given a set of images I with a target segmentation $X^*(I)$.

3.3 COST FUNCTION FOR LEARNING

Definitions $X_p[W]$, λ_p are the p^{th} largest eigenvector, eigenvalue of $WX = \lambda D_W X$ with $\|X_p[W]\| = 1$ and $D_W = diag(W\mathbf{1})$. $X_p[W]$ is uniquely defined up to polarity, which we disambiguate using a fixed vector Y and forcing $\text{sign}(Y^T X_p[W]) = +1$. This is possible only when $Y^T X_p[W] \neq 0$. We also require λ_p be unique. To satisfy these constraints, we will restrict our attention to weight matrices W in a certain subset $S_n^{2,X^*(I)}$ of symmetric matrices, where $S_n^{p,Y} = \{W \in S_n : W\mathbf{1} > 0, \ker(W - \lambda_p D_W) \not\subset Y^\perp,$ λ_p single$\}$. Note that if $W \in S_n$ and $W\mathbf{1} > 0$, W has probability 1 of being in the feasible space. What's more, $S_n^{2,X^*(I)}$ is open, which implies that any small perturbation of W is allowed.
Define the one-target energy function:

$$\mathcal{E}(W,I) = \frac{1}{2}\|X_2[W(I)] - X^*(I)\|^2, \text{ for } W \in S_n^{2,X^*(I)} \tag{3}$$

The multi-target energy function is defined as $\mathcal{E}(W) = \sum_I \mathcal{E}(W,I)$, for $W \in \cap_I S_n^{2,X^*(I)}$. This error energy function has the following property, which will be useful later on when we try to learn the graph network.

Prop. 3.1 ($\mathcal{E}(W,I)$ has no local minimum) *The single target energy function has all its local minima in $S_n^{2,X^*} \bigcap \{W : \lambda_2(W) \neq -1\}$ equal to the global minimum, 0.*

The proof, in [2], shows that at a critical point, the error vector $X_2 - X^*(I)$ is in the kernel of a certain matrix of rank n-1. This shows in fact that $X_2 - X^*(I)$ is proportional to X_2, which finally leads to $X_2 = X^*(I)$.

3.4 GRADIENT DESCENT ALGORITHM

We minimize the error energy over W by gradient descent: $\Delta W = -\eta \frac{\partial \mathcal{E}}{\partial W} = -\eta \frac{\partial \mathcal{E}}{\partial X_2} \frac{\partial X_2}{\partial W}$. When W is parameterized by Θ, we have instead $\Delta\Theta =$

$-\eta \frac{\partial \mathcal{E}}{\partial X_2} \frac{\partial X_2}{\partial W} \frac{\partial W}{\partial \Theta}$. In the case of rectangle detection, the parameters Θ consist of all the values of the function $f(e_i, e_j) = \bar{f}(x_2 - x_1, y_2 - y_1, \theta_1, \theta_2)$, which is a 4 dimensional lookup table.

The main difficulty is to study how the Ncut eigenvector, $X_2[W(\Theta)]$, varies with the graph weight matrix $W(\Theta)$. We will write down a continuous-time PDE describing evolution of the error energy on $X_2[W(\Theta)]$ with respect to Θ. We show that this PDE has an *exact analytical* form, and the resulting PDE *converges*. We have also proved the convergence rate is exponential for a simple case[2]. This result shows that we can minimize the error energy over W or Θ by gradient descent.

Theorem 3.2 (Derivative of Ncut eigenvector) *The map $W \rightarrow (X_p, \lambda_p)$ is C^∞ over $S_n^{p,Y}$, and we can express the derivatives over any C^1 path $W(t)$ as:*

$$\begin{aligned} \frac{dX_p[W(t)]}{dt} &= -(W - \lambda_p D_W)^\dagger \\ &\quad (W' - \lambda_p D'_W - \frac{d\lambda_p}{dt} D_W) X_p \\ \frac{d\lambda_p}{dt} &= \frac{X_p^T (W' - \lambda_p D'_W) X_p}{X_p^T D_W X_p} \end{aligned}$$

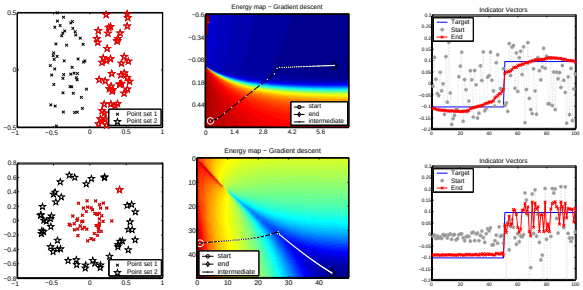
We obtain an analog theorem for the **derivative of standard eigenvectors**, by simply replacing D_W with I_n. The proof in [2] uses the implicit function theorem to show $X_p[W]$ is C^∞, then differentiates $WX_p = \lambda_p D_W X_p$ to obtain $(W - \lambda_p D_W)X'_p + (W' - \lambda_p D'_W - \lambda'_p D_W)X_p = 0$.

Computation of the partial derivatives $\frac{\partial X_2}{\partial W}$ alone requires $O(n^3)$ time because of the pseudo-inverse term $(W - \lambda_p D_W)^\dagger$ in each gradient direction. We remove this bottleneck by first left-multiplying by $\frac{\partial \mathcal{E}}{\partial X_2}$. We introduce $Y = -(W - \lambda_2 D_W)^\dagger (X_2 - X^*(I))$, which we showed how to compute efficiently in [2], and obtain a $O(n^2)$ gradient update rule:

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial W}_{ij} &= X_{2,i} Y_j + X_{2,j} Y_i - \lambda_2 (X_{2,i} Y_i + X_{2,j} Y_j) \\ &\quad -\lambda'_{2ij} Y^T D_W X_2 \\ \text{with } \lambda'_{2ij} &= \frac{2X_{2,i}X_{2,j} - \lambda_2 ({X_{2,i}}^2 + {X_{2,j}}^2)}{X_2^T D_W X_2} \end{aligned}$$

3.5 PROPERTIES OF THE LEARNING ALGORITHM

Empirically, we observe that $\mathcal{E}(W(t))$ converges to 0 exponentially fast when $W(t)$ follows the gradient path, even if the number of training examples grows as $O(n)$. We will prove this fact in the case of a single target. The convergence of $\mathcal{E}(W(t))$ however does not imply that of $W(t)$. Indeed, one can construct functions for which gradient descent leads to limit cycle oscillations. The following proposition shows that this cannot happen here.



(A) Point set (B) Error Energy (C) Ncut eigenvector

Figure 6: Learning point set clustering. $W(i,j) = \exp(-\sigma_x(x(i) - x(j))^2) + \exp(-\sigma_y(y(i) - y(j))^2)$. A) 2D layout of the points. The first set is the cross set and the second is the star set. The resulting clustering can be identified by the red and black colors. B) Energy landscape of $\mathcal{E}(\sigma_x, \sigma_y)$, and gradient path taken by Eq.4, $(\dot{\sigma_x}, \dot{\sigma_y}) = -(\frac{\partial \mathcal{E}}{\partial \sigma_x}, \frac{\partial \mathcal{E}}{\partial \sigma_y})$. C) Target vector comparing with initial and final learned Ncut vector. The graph nodes are ordered according to their x-axis position (first row), and to their distance to origin (second row).

Prop. 3.3 (Exponential convergence of $\mathcal{E}(W, I)$) *The 1-target energy PDE $\dot{W} = -\frac{\partial \mathcal{E}}{\partial W}$ either converges to a global energy minimum W_∞, or it escapes any compact $K \subset S_n^{2,X^*}$. In the first case, $\mathcal{E}(W(t)) \rightarrow 0$ exponentially.*

Our proof in [2] shows that $\|\frac{\partial \mathcal{E}}{\partial W}\| \geq b\sqrt{\mathcal{E}}$, leading to the convergence of $W(t)$, and then $\frac{d}{dt}\mathcal{E}(W(t)) \leq -b^2\mathcal{E}$, which shows the exponential decay of $\mathcal{E}(W(t))$.

Pathological non-convergence cases. As stated in the proposition, $W(t)$ could potentially hit the boundary of S_n^{2,X^*}. This arises in 2 pathological cases: 1) $\lambda_2(t) \rightarrow 1$ or $\lambda_2(t) - \lambda_3(t) \rightarrow 0$, and 2) $D_{W(t)}(i,i) \rightarrow 0$ for some i. Note that $X^*(I)^T X_2[W(t)] \rightarrow 0$ cannot happen, because initially $X^*(I)^T X_2[W(t)] > 0$ and $\mathcal{E}(W, I)$ decreases. There are ways to alleviate those problems through weight parameterization, but in practice they only occur when learning a lot of target vectors.

4 RESULTS

4.1 POINT SET CLUSTERING

In experiment 1, figure 6, we examine our spectral graph learning algorithm on simple 2D point set clustering examples. The graph weight matrix $W_{ij} =$

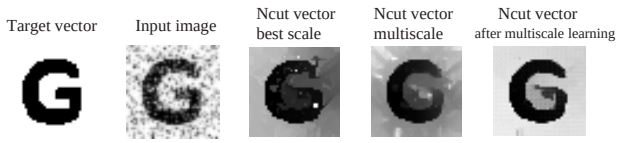


Figure 7: A simple example of multiscale learning. The input image is 40 by 40 and 4 narrow scales are used. The best result (minimum of energy) with one scale is displayed as well as the result with all four scales set up with the same weight, and with the learned weight. The use of multiscale enable to segment correctly the inside of the G. The lowest energy is achieved after learning.

$\exp(-\sigma_x(x_i - x_j)^2) + \exp(-\sigma_y(y_i - y_j)^2)$ has two parameters σ_x, σ_y which we aim to optimize. We update (σ_x, σ_y) as follows:

$$\Delta\sigma_x = -\eta(X - X^*)^T \frac{\partial X}{\partial \sigma_x} \quad (4)$$

$$\Delta\sigma_y = -\eta(X - X^*)^T \frac{\partial X}{\partial \sigma_y} \quad (5)$$

We use directly the derivatives given in Sec. 3.4 with the following expressions of W':

$$\frac{\partial w_{ij}}{\partial \sigma_x} = -(x_i - x_j)^2 \, e^{-\sigma_x (x_i - x_j)^2} \quad (6)$$

$$\frac{\partial w_{ij}}{\partial \sigma_y} = -(y_i - y_j)^2 \, e^{-\sigma_y (y_i - y_j)^2} \quad (7)$$

The experiments on simple clustering show a fast convergence of the gradient descent. We also tested the algorithm with radial distributed point sets.

4.2 MULTISCALE IMAGE SEGMENTATION

In this experiment, we focus on an application of spectral learning in image segmentation. The aim is to provide a powerful tool to find the best scales of edge extraction in Ncut segmentation[3]. Basically the idea of multiscale segmentation is to use several edge scales find a consistent segmentation of the image across scales. The simultaneous use of various scale levels is interesting for complex and big images which mixes textures with sharp and soft contours. In those images, meaningful boundaries may exist at weak contours or between textures that do not rise to edges. Using simultaneously several scales of edges enable to face this problem. The global affinity matrix is the sum of r-affinity matrices at different scales:

$$W_{i,j}^{(I)} = \sum_{r=1}^{k} \alpha_r \exp\left(-\sigma_r \Delta_{i,j}^{(r)}(I)\right) \quad (8)$$

where $\Delta_{i,j}^{(r)}(I)$ is a matrix of the same size as W which expresses a distance measure in a specific scale. Learning on the α coefficients of the scales enables to select

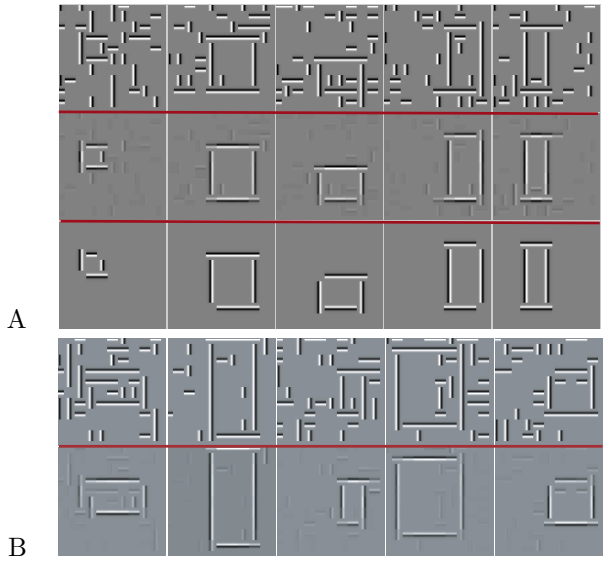


Figure 8: A: Training. Row 1: training input vector. Row 2: Ncut vector after learning. Row 3: target vector. B: Testing. Row 1: testing input vector, Row 2: Ncut vector after learning. For each edge, 2 polarities are hypothesized (only 1 is displayed in Row 1 of Training/Testing). Notice that after learning, not only clutter edges are suppressed but also the correct edge polarities are recovered.

the scales and to set up the weighting coefficients when more than two scales are required. Learning on σ coefficients enables to find the sensitivity to edge strength at a given scale.

The update rules for α_r, σ_r are as following:

$$\Delta\alpha_r = -\eta Y\left(\frac{\partial W}{\partial \alpha_r} - \lambda \frac{\partial D}{\partial \alpha_r} - \frac{\partial \lambda}{\partial \alpha_r} D\right) X \quad (9)$$

$$\Delta\sigma_r = -\eta Y\left(\frac{\partial W}{\partial \sigma_r} - \lambda \frac{\partial D}{\partial \sigma_r} - \frac{\partial \lambda}{\partial \sigma_r} D\right) X \quad (10)$$

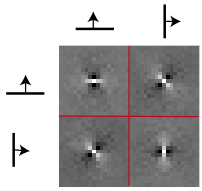


Figure 9: The learned shift-invariant graph clique function $\bar{f}(x_2 - x_1, y_2 - y_1, \theta_1, \theta_2)$ with $\theta_1 = 0$, and $\theta_2 = \pi/2$. Each 2D function corresponds to a fixed (θ_1, θ_2) pair. The clique function learns to favor good continuation of the edges with $(\theta_1 = 0, \theta_2 = 0)$, $(\theta_1 = \pi/2, \theta_2 = \pi/2)$, and corner configurations $(\theta_1 = \pi/2, \theta_2 = 0)$, $(\theta_1 = 0, \theta_2 = \pi/2)$

4.3 SHAPE DETECTION

We first generate random rectangles in synthetic images of 100 by 100, see Fig. 8. The edges extracted, e_i are specified by its quantized location (x_i, y_i), orientation θ_i, and polarity p_i. Three graph weight clique potential functions are implemented:

1. unconstrained $f(e_1, e_2) = f(x_1, y_1, \theta_1, x_2, y_2, \theta_2)$
2. translational invariant $f(e_1, e_2) = \bar{f}(x_2 - x_1, y_2 - y_1, \theta_1, \theta_2)$
3. translational invariant with ternary clique potential $f(e_1, e_2) = \frac{1}{N} \sum_k g(x_1 - x_k, y_1 - y_k, x_2 - x_k, y_2 - y_k, \theta_1, \theta_2, \theta_k)$.

We apply the following graph learning algorithms to train segmentation algorithm to detect rectangles.

1. Generate random rectangles with one noise-free and one noisy version per example. Generate a random affinity matrix W to start with
2. For each image I, extract edge features from the noisy image to compute subgraph $V(I)$, and compute target $X^*(I)$ from the noise-free image
3. initialize $\mathcal{E} = 0$; for each image I,
   (a) $W(I) = W(\mathrm{V}(I), \mathrm{V}(I))$, using one of the clique potential function $f(e_i, e_j)$ described above.
   (b) Compute $X_2(I)$, second generalized eigenvector of $(W(I), D_{W(I)})$
   (c) Update $W(I)$ with gradient update and propagate updating to each $f(e_i, e_j)$
   (d) Update $\mathcal{E} := \mathcal{E} + \mathcal{E}_I$ with the partial energy $\mathcal{E}_I = \frac{1}{2}||X_2(I) - X^*(I)||^2$
4. Go back to step 3 until $\mathcal{E} < threshold$

Fig.8 display the results of the training and testing using shift-invariant clique function $\bar{f}$. Figure 4.2 shows the shift-invariant clique function learned on a pair of horizontal and vertical edges.

4.4 COMPARISON BETWEEN THE DIFFERENT CLIQUE POTENTIAL FUNCTIONS

We have applied the three methods to random rectangles in 100 by 100 images. The unconstrained affinity matrix has 40000 entries, shift-invariant clique function $\bar{f}$ has 1444 entries and triplet clique function g has 1042568 entries. Several simulations have been run

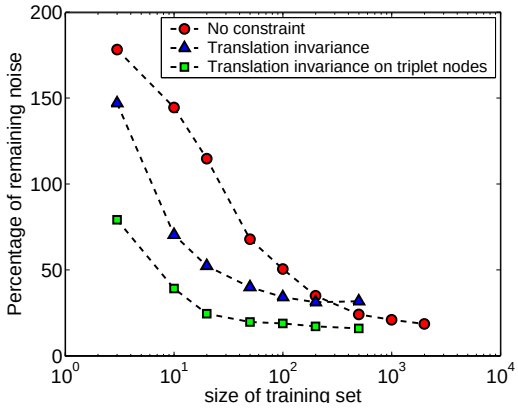


Figure 10: Square shape detection and enhancement. For each simulation, the table indicates the percentage of remaining noise (100 is the initial amount of noise) on 2000 testing examples. These results have been obtained for different sizes of training set, according to three methods: 1- learning on full affinity matrix, 2- invariance by translation, 3- mean on third node with invariance by translation.

for each training set and the result displayed in fig.10 have been averaged. We noticed a very low standard deviation on our training sets.

We see that the best results are achieved with the triplet clique method involving summation of ternary affinities over a third node. With only 20 training examples, an average of 75% of the noise was eliminated in the 2000 testing examples. We achieved the best result with a training data set of 500 squares. 15 iterations were enough to reach energy convergence and it took 4 minutes. This fast convergence can be explained by the averaging on a third node: when the affinity between two nodes is updated, all ternary affinities involving this pair are updated in a single pass. Also, ternary clique potentials carry out a stronger, more robust cue than binary affinities.

4.5 RECTANGLE DETECTION ON REAL IMAGES

This rectangle detection algorithm can be applied directly on real images, fig.12. We just have to adapt the filter parameters to have a good edge extraction. The amount of noise on real images turns out to be frequently below the one we used in the learning step, thus giving those encouraging results.

References

[1] Francis R. Bach and Michael I. Jordan. Learning spectral clustering. *Advances in Neural Information Processing Systems (NIPS)*, 2003.

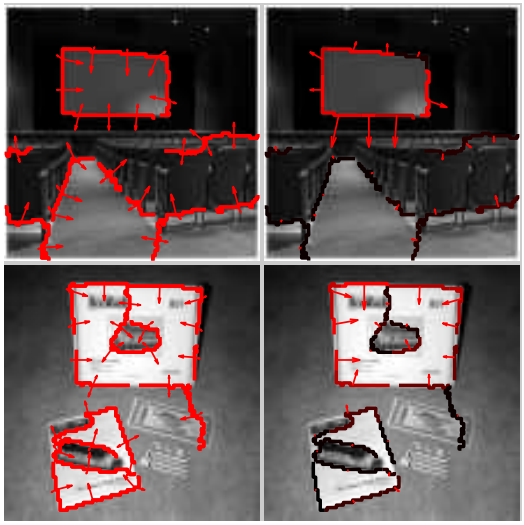


Figure 11: Examples of rectangle detection on real images. Left: edges detected, with arrows indicating orientation (2 opposite polarity hypothesis for each edge). Right: segmentation of foreground edges (in red) versus background clutter edges (in dark).

[2] Timothee Cour and Jianbo Shi. A learnable spectral memory graph for recognition and segmentation. Technical Report MS-CIS-04-12, University of Pennsylvania CIS Technical Reports, Philadelphia, PA, June 2004.

[3] Charless Fowlkes, David Martin, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence(PAMI)*, 26(5):530–549, 2004.

[4] Marina Meila and Jianbo Shi. Learning segmentation with random walk. *Advances in Neural Information Processing Systems (NIPS)*, 2001.

[5] Andrew Y. Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[6] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence(PAMI)*, 22(8):888–905, 2000.

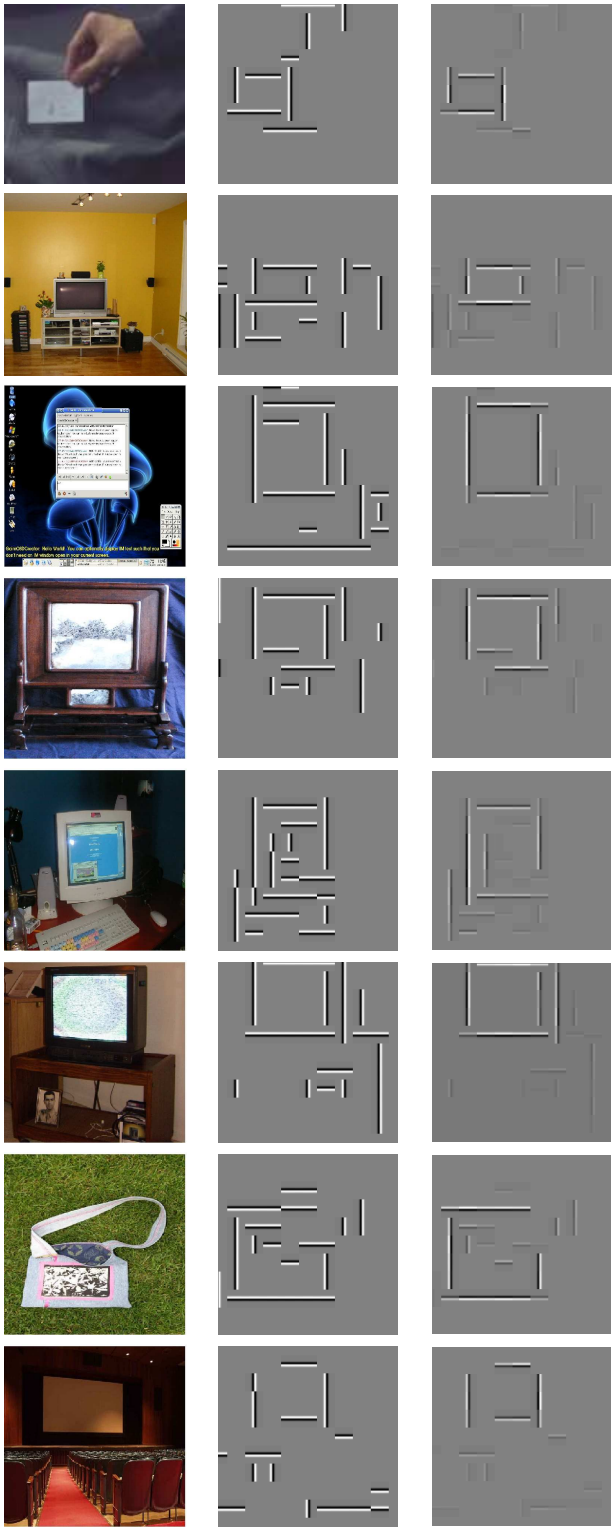


Figure 12: Rectangle detection on real images. First column: image; second column: edges detected; third column: rectangle detection using the graph. Graph weights are learned with random rectangles in background noise.

A Graphical Model for Simultaneous Partitioning and Labeling

Philip J. Cowans
Cavendish Laboratory, University of Cambridge,
Cambridge, CB3 0HE, United Kingdom
pjc51@cam.ac.uk

Martin Szummer
Microsoft Research
Cambridge, CB3 0FB, United Kingdom
szummer@microsoft.com

Abstract

In this work we develop a graphical model for describing probability distributions over labeled partitions of an undirected graph which are conditioned on observed data. We show how to efficiently perform exact inference in these models, by exploiting the structure of the graph and adapting the sum-product and max-product algorithms. We demonstrate our approach on the task of segmenting and labeling hand-drawn ink fragments, and show that a significant performance increase is obtained by labeling and partitioning simultaneously.

1 INTRODUCTION

Probabilistic models are usually defined over the Cartesian product of a number of discrete or continuous one-dimensional spaces. For example, models performing joint binary classification of N objects are defined over $\{-1,+1\}^N$. While in many cases it is intractable to explicitly enumerate all possible configurations, in the case of graphical models where the probability distribution factors according to the structure of an undirected graph, message passing techniques such as the sum-product and max-product algorithms can be used to render the computation feasible.

In this work, we extend the graphical model formalism to the case of probability distributions defined over labeled partitions of an undirected graph; in other words, possible divisions of the graph into sets of vertices referred to as *parts*, where each part is assigned a label. An example of a labeled partition is given in Figure 1. Note that the number of parts varies between partitions and is usually unknown in advance. Our method represents partitions directly, rather than incorporating part identifiers into the labels. We thereby avoid the degeneracy that different permutations of part identifiers represent the same partition (see Section 3.4 for a comparison of the two approaches). In this work we restrict ourselves to binary labels, but the method can be generalized straightforwardly to larger label sets. Conversely, unlabeled partitioning may be viewed as a special case with just one label. Our model is similar to the Conditional Random Field (CRF) [2], and allows the probability distribution to be conditioned on arbitrary observed data. This model is widely applicable to joint segmentation and classification tasks, which are common in computer vision, handwriting recognition, speech and natural language processing. The Markov Random Field (MRF), which is an undirected graphical model whose potential functions do not depend on observed data, is for the purposes of this paper a special case of the CRF, and can also be extended in the way described below.

Previously, probabilistic models have been used for graph partitioning, but by using Monte Carlo techniques rather than exact inference [1]. Liu [4] has performed partitioning, but not using a probabilistic framework. Other work [7] has extended the CRF to perform multiple inference tasks simultaneously, but has not considered partitioning of non-linear graphs.

We begin by describing the full probabilistic model, then consider representations for labeled partitions and efficient algorithms for performing the necessary

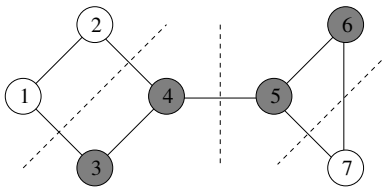


Figure 1: An example of a labeled partition. Vertices are partitioned as follows: $(1,2,+)$, $(3,4,-)$, $(5,6,-)$, $(7,+)$, where the last symbol in each group indicates the label assigned to that part.

inference tasks. Finally, we describe the application of our model to the task of parsing hand-drawn ink diagrams.

2 THE PROBABILISTIC MODEL

Let $\mathcal{G}$ be an undirected graph consisting of vertices $\mathcal{V}$ and edges $\mathcal{E}$. We assume that $\mathcal{G}$ is triangulated, so that every cycle of length greater than three is spanned by a chord. This can always be achieved by adding edges, but usually at the expense of increasing the maximum clique size, and therefore computational complexity.

Let S be a partition of $\mathcal{G}$, that is, a set of non-empty subsets of $\mathcal{V}$, such that each vertex in $\mathcal{V}$ is a member of precisely one subset. Each subset is referred to as a *part* of $\mathcal{G}$. In this paper, the term *partition* will always refer to a *contiguous* partition:

Definition 1. *A partition of $\mathcal{G}$ is **contiguous** if and only if all parts are internally connected. In other words, if i and j are vertices contained within the same part, there exists a path on $\mathcal{G}$ between i and j entirely contained within that part.*

A labeled partition of $\mathcal{G}$ is represented by $\mathcal{Y} = (\mathsf{S}, \boldsymbol{y})$, where S describes the partition and $\boldsymbol{y} \in \{-1, +1\}^M$ is a vector containing the labels associated with each part. For example, a partition of three elements into two parts could be $\mathsf{S} = \{\{1\}\{2,3\}\}, \boldsymbol{y} = [+1, -1]$. Let $\mathbb{Y}$ be the set of all possible labeled partitions of $\mathcal{G}$. Note that M, the length of $\boldsymbol{y}$, is dependent on S. Let t_i be the index of the part to which vertex i is assigned, so that y_{t_i} is the label given to that vertex.

In this work, the conditional probability distribution over $\mathbb{Y}$ has the form $P(\mathcal{Y} \mid \boldsymbol{x}, \boldsymbol{\theta}) =$

$$\frac{1}{Z(\boldsymbol{\theta})} \prod_{i \in \mathcal{V}} \psi_i^{(1)}(\mathcal{Y}, \boldsymbol{x}; \boldsymbol{\theta}) \prod_{i,j \in \mathcal{E}} \psi_{ij}^{(2)}(\mathcal{Y}, \boldsymbol{x}; \boldsymbol{\theta}), \quad (1)$$

where $\boldsymbol{x}$ is the observed data, $\boldsymbol{\theta}$ is a vector representing the model parameters collectively, and $Z(\boldsymbol{\theta})$ is a normalization constant. $\psi_i^{(1)}$ are unary potentials defined for each vertex, and $\psi_{ij}^{(2)}$ are pairwise potentials defined for each edge. The unary potentials introduce a data-dependent bias towards assigning one label or the other to each vertex. The pairwise potentials model the compatibility between the parts and labels of neighboring vertices, and are also data dependent. The dependence of these potentials on $\boldsymbol{x}$ is through feature vectors, $\boldsymbol{g}_i$ and $\boldsymbol{f}_{ij}$, defined for each vertex i and edge (i,j) respectively. The potentials then have the form

$$\psi_i^{(1)}(\mathcal{Y}, \boldsymbol{x}, \boldsymbol{\theta}) = \begin{cases} \phi(\boldsymbol{w}_+ \cdot \boldsymbol{g}_i(\boldsymbol{x})) & \text{if } y_{t_i} = +1 \\ \phi(\boldsymbol{w}_- \cdot \boldsymbol{g}_i(\boldsymbol{x})) & \text{if } y_{t_i} = -1 \end{cases}, \quad (2)$$

where $\phi(\cdot)$ is a non-linear mapping, and $\boldsymbol{w}_+$ and $\boldsymbol{w}_-$ are vectors of feature weights depending on the label of the appropriate vertex. In this work, we will always use an exponential non-linearity, $\phi : x \mapsto \exp(x)$, although in general other functions may be used. The pairwise potentials are defined by

$$\psi_{ij}^{(2)}(\mathcal{Y}, \boldsymbol{x}, \boldsymbol{\theta}) = \begin{cases} \phi(\boldsymbol{v}_{\text{ss}} \cdot \boldsymbol{f}_{ij}(\boldsymbol{x})) & \text{if } t_i = t_j,\ y_{t_i} = y_{t_j} \\ \phi(\boldsymbol{v}_{\text{sd}} \cdot \boldsymbol{f}_{ij}(\boldsymbol{x})) & \text{if } t_i \neq t_j,\ y_{t_i} = y_{t_j} \\ \phi(\boldsymbol{v}_{\text{dd}} \cdot \boldsymbol{f}_{ij}(\boldsymbol{x})) & \text{if } t_i \neq t_j,\ y_{t_i} \neq y_{t_j} \end{cases} \quad (3)$$

where $\boldsymbol{v}_{\text{ss}}$, $\boldsymbol{v}_{\text{sd}}$ and $\boldsymbol{v}_{\text{dd}}$ are vectors of feature weights to be used when i and j belong to the same part, different parts with the same label, and different parts with different labels respectively. The fourth case, corresponding to vertices with different labels in the same part, does not occur by definition. The parameters in $\boldsymbol{\theta}$ are therefore $(\boldsymbol{w}_+, \boldsymbol{w}_-, \boldsymbol{v}_{\text{ss}}, \boldsymbol{v}_{\text{sd}}, \boldsymbol{v}_{\text{dd}})$. Note that there is a redundancy in the weight vectors. In practice, $\boldsymbol{w}_-$ and $\boldsymbol{v}_{\text{dd}}$ were constrained to be $\mathbf{0}$.

2.1 TRAINING

The overall goal of the model above is to predict labeled partitions of unseen data. In order to do this, we must first estimate the model parameters, $\boldsymbol{\theta}$. These parameters are learned from example data. Given a labeled training examples, $(\boldsymbol{x}, \mathcal{Y})$, the posterior probability of the parameters is given using Bayes' rule,

$$P(\boldsymbol{\theta} \mid \boldsymbol{x}, \mathcal{Y}) \propto P(\mathcal{Y} \mid \boldsymbol{x}, \boldsymbol{\theta}) \cdot P(\boldsymbol{\theta}), \quad (4)$$

where $P(\boldsymbol{\theta})$ is a prior distribution over the weights. The model is trained by finding the maximum *a posteriori* weights using a quasi-Newton gradient ascent algorithm (specifically, the BFGS method). A significant advantage is that the model is convex in the parameters, meaning that we are guaranteed to find the global maximum using gradient ascent. The gradient of the log posterior, $\mathcal{LP}$, with respect to a parameter θ_k is given by

$$\frac{\partial}{\partial \theta_k} \mathcal{LP} = \sum_{i \in \mathcal{V}} \left(\frac{\partial}{\partial \theta_k} \log \psi_i^{(1)} - \left\langle \frac{\partial}{\partial \theta_k} \log \psi_i^{(1)} \right\rangle \right) + \frac{\partial}{\partial \theta_k} \log(\mathrm{P}(\boldsymbol{\theta})) \quad (5)$$

if θ_k is a parameter of the unary potentials. The gradients with respect to parameters of the pairwise potentials have a similar form. It is straigtforward to generalize this expression to handle multiple training examples. The brackets, $\langle \cdots \rangle$, in the second terms represent expectations with respect to the distribution over $\mathbb{Y}$ given by the current parameter values. This requires the computation of marginal probability distributions for individual vertices and pairs of vertices connected

by an edge. Furthermore, the optimization algorithm needs to evaluate (1) explicitly, which in turn requires evaluation of the partition function,

$$Z(\boldsymbol{\theta}) = \sum_{\mathcal{Y}} \prod_{i \in \mathcal{V}} \psi_i^{(1)}(\mathcal{Y}, \boldsymbol{x}; \boldsymbol{\theta}) \prod_{i,j \in \mathcal{E}} \psi_{ij}^{(2)}(\mathcal{Y}, \boldsymbol{x}; \boldsymbol{\theta}). \quad (6)$$

Both of these tasks involve summations over subsets of possible labeled partitions. This summation can be performed efficiently by message passing using a modified version of the sum-product algorithm. The details of this algorithm will be given in Section 3 below.

2.2 INFERENCE

In general, we are interested in using the trained model to group and label unseen data. This is achieved by finding the most probable configuration,

$$\mathcal{Y}^{\mathrm{MAX}} = \arg\max_{\mathcal{Y}} \prod_{i \in \mathcal{V}} \psi_i^{(1)}(\mathcal{Y}, \boldsymbol{x}; \boldsymbol{\theta}) \prod_{i,j \in \mathcal{E}} \psi_{ij}^{(2)}(\mathcal{Y}, \boldsymbol{x}; \boldsymbol{\theta}). \quad (7)$$

As with the partition function, this maximization can be performed efficiently using a version of the max-product algorithm.

3 OPERATIONS OVER LABELED PARTITIONS

In Section 2, it was shown that an important part of both the training and inference processes is the enumeration of all possible labeled partitions, in order to either sum or maximize over them. As in the more usual case of labeling vertices, explicit enumeration of all possible values is prohibitively expensive. However, as we show below, we are able to exploit the structure of the graph to significantly reduce the computational cost, rendering exact inference tractable in many cases. The derivation below follows the conditions for the possibility of local computation provided by Shenoy and Shafer [5]. An alternative derivation however is possible following Lauritzen [3].

If G is a subset of $\mathcal{V}$, we use $\mathcal{Y}_G \in \mathbb{Y}_G$ to denote a labeled partition of the corresponding induced subgraph. We define *consistency* as follows:

Definition 2. *Labeled partitions $\mathcal{Y}_G$ and $\mathcal{Y}_H$, of subgraphs G and H respectively, are **consistent**, denoted $\mathcal{Y}_H \backsim \mathcal{Y}_G$, if and only if:*

1. *All vertices appearing in $G \cap H$, are assigned the same label by $\mathcal{Y}_G$ and $\mathcal{Y}_H$, and*
2. *All pairs of vertices appearing in $G \cap H$ are in the same part in $\mathcal{Y}_G$ if and only if they are in the same part in $\mathcal{Y}_H$.*

The notation $\hat{\mathcal{Y}}_G(\mathcal{Y}_{G \cup H})$ is used to denote the unique labeled partition of G which is consistent with $\mathcal{Y}_{G \cup H}$. The maximal cliques of $\mathcal{G}$ are defined in the usual way, and are denoted $C_1, \ldots, C_N$. If b and t are two cliques, and b contains all vertices from t which appear in cliques other than t, then b is said to be a *branch* and t is the corresponding *twig*.

Following the framework of Shenoy and Shafer, we introduce the notion of a *valuation* ψ on a subset of $\mathcal{V}$. In the case of standard belief propagation, valuations are functions assigning a real, non-negative value to possible configurations of subsets of the variables. In this work, a valuation on a subset G will be defined as a function mapping $\mathbb{Y}_G$ to the non-negative real numbers. $\mathbb{V}_G$ is the set of all valuations on G. In the case where the valuation is over the whole of $\mathcal{G}$, the range of the valuation will be interpreted as being proportional the probability of the corresponding labeled partition. In the case of valuations defined over subsets of $\mathcal{V}$ the valuations are referred to as potentials of which those defined in (1) are an example. We define two operations on valuations:

1. **Combination:** Suppose G and H are subsets of $\mathcal{V}$ and ψ_G and ψ_H are valuations on those subsets. The operation of combination defines a mapping $\otimes : \mathbb{V}_G \times \mathbb{V}_H \mapsto \mathbb{V}_{G \cup H}$, such that

$$\psi_G \otimes \psi_H(\mathcal{Y}_{G \cup H}) \triangleq \psi_G\left(\hat{\mathcal{Y}}_G(\mathcal{Y}_{G \cup H})\right) \cdot \psi_H\left(\hat{\mathcal{Y}}_H(\mathcal{Y}_{G \cup H})\right). \quad (8)$$

2. **Marginalization:** Suppose G and H are subsets of $\mathcal{V}$ such that $G \subseteq H$, and ψ_G and ψ_H are valuations as before. Marginalization is a mapping $\downarrow : \mathbb{V}_H \mapsto \mathbb{V}_G$ such that

$$\psi_H^{\downarrow G}(\mathcal{Y}_G) \triangleq \sum_{\mathcal{Y}_H \backsim \mathcal{Y}_G} \psi_H(\mathcal{Y}_H). \quad (9)$$

A valuation over the whole graph is said to factor if it can be written as the combination of valuations on the cliques,

$$\psi(\mathcal{Y}) = \bigotimes_{i=1}^{N} \psi_i(\mathcal{Y}_i), \quad (10)$$

where i runs over the cliques in $\mathcal{G}$. As combination allows products of valuations over subsets of a clique to be written in terms of a single valuation over the whole clique, the model given in (1), excluding the partition function, is in this form. Before demonstrating the possibility of efficient local computation, we first demonstrate that three axioms due to Shenoy and Shafer are satisfied:

Axiom 1. Commutativity and associativity of combination. *If G, H and K are subsets of $\mathcal{V}$, for any valuations ψ_G, ψ_H and ψ_K, we have $\psi_G \otimes \psi_H = \psi_H \otimes \psi_G$ and $\psi_G \otimes (\psi_H \otimes \psi_K) = (\psi_G \otimes \psi_H) \otimes \psi_K$.*

Proof. Follows directly from the definition of combination. □

Axiom 2. Consonance of marginalization, *If G, H and K are subsets of $\mathcal{V}$ such that $K \subseteq G \subseteq H$, for any valuations ψ_G, ψ_H and ψ_K,*

$$\left(\psi_H^{\downarrow G}\right)^{\downarrow K} = \psi_H^{\downarrow K}. \tag{11}$$

Proof. Writing the marginalization explicitly,

$$\begin{aligned}\left(\psi_H^{\downarrow G}\right)^{\downarrow K} &= \sum_{\mathcal{Y}_G \backsim \mathcal{Y}_K} \sum_{\mathcal{Y}_H \backsim \mathcal{Y}_G} \psi_H(\mathcal{Y}_H) \\ &= \sum_{\mathcal{Y}_H \backsim \mathcal{Y}_K} \psi_H(\mathcal{Y}_H) = \psi_H^{\downarrow K},\end{aligned} \tag{12}$$

where the second line follows as for any $\mathcal{Y}_H \backsim \mathcal{Y}_K$ there is a unique $\mathcal{Y}_G$ such that $\mathcal{Y}_G \backsim \mathcal{Y}_K$ and $\mathcal{Y}_H \backsim \mathcal{Y}_G$, and for any $\mathcal{Y}_H \not\backsim \mathcal{Y}_K$, no such $\mathcal{Y}_G$ exists. □

Axiom 3. Distributivity of marginalization over combination, *If G and H are subsets of $\mathcal{V}$, for any valuations ψ_G and ψ_H, $(\psi_G \otimes \psi_H)^{\downarrow G} = \psi_G \otimes (\psi_H^{\downarrow G \cap H})$.*

Proof. Performing an explicit expansion gives

$$\begin{aligned}(\psi_G \otimes \psi_H)^{\downarrow G} &= \sum_{\mathcal{Y}_{G\cup H} \backsim \mathcal{Y}_G} \psi_G\left(\hat{\mathcal{Y}}_G(\mathcal{Y}_{G\cup H})\right) \cdot \\ &\qquad \psi_H\left(\hat{\mathcal{Y}}_H(\mathcal{Y}_{G\cup H})\right) \\ &= \psi_G(\mathcal{Y}_G) \cdot \sum_{\mathcal{Y}_{G\cup H} \backsim \mathcal{Y}_G} \psi_H\left(\hat{\mathcal{Y}}_H(\mathcal{Y}_{G\cup H})\right) \\ &= \psi_G(\mathcal{Y}_G) \cdot \sum_{\mathcal{Y}_H \backsim \hat{\mathcal{Y}}_{G\cap H}(\mathcal{Y}_G)} \psi_H(\mathcal{Y}_H),\end{aligned} \tag{13}$$

which is equal to $\psi_G \otimes (\psi_H^{\downarrow G\cap H})$ by definition. □

3.1 THE SUM-PRODUCT ALGORITHM

In the next two sections we develop an extension of the sum-product algorithm suitable for probability distributions over partitions. As with the more usual form of this algorithm, our method exploits the known structure of $\mathcal{G}$ by passing messages containing the results of local computations. Our goal is to compute sums over a subset of all possible partitions, such as those needed for the partition function, as given in (6). This task should be contrasted with that of the usual sum-product algorithm [3], which sums over assignments of labels to the vertices. Since we sum over a different domain we will need to modify the messages passed and the ranges of summation. Later, in Section 3.3, we will also adapt the max-product algorithm for labeled partitions. Consider a sum of form:

$$f_s(\mathcal{Y}_1) = \sum_{\mathcal{Y} \backsim \mathcal{Y}_1} P^*(\mathcal{Y}) = \left(P^*(\mathcal{Y})\right)^{\downarrow C_1}, \tag{14}$$

where P^* is a (possibly unnormalized) probability distribution[1] over labeled partitions of $\mathcal{G}$. Let the cliques be numbered $C_1, \ldots, C_N$, such that C_1 is the clique containing the vertices onto which we wish to marginalize and such that for all k, C_k is a twig in the graph $C_1 \cup C_2 \cup \ldots \cup C_k$. Such an ordering is always possible if $\mathcal{G}$ is triangulated. According to Axiom 2, this can be expressed as

$$\begin{aligned}f_s(\mathcal{Y}_1) &= \left(\left(P^*(\mathcal{Y})\right)^{\downarrow \mathcal{V}\setminus C_N}\right)^{\downarrow C_1} \\ &= \left(\left(\bigotimes_{i=1}^{N} \psi_i(\mathcal{Y}_i)\right)^{\downarrow \mathcal{V}\setminus C_N}\right)^{\downarrow C_1} \\ &= \left(\left(\bigotimes_{i=1}^{N-1} \psi_i(\mathcal{Y}_i)\right) \otimes \left(\psi_N(\mathcal{Y}_N)^{\downarrow C_N \cap \mathcal{V}}\right)\right)^{\downarrow C_1}.\end{aligned} \tag{15}$$

In the last step, Axiom 3 has been used. C_N is a twig by construction. Let C_B be a corresponding branch, then $C_N \cap \mathcal{V} = C_N \cap C_B$, hence

$$\begin{aligned}f_s(\mathcal{Y}_1) = \Bigg(\Bigg(\bigotimes_{\substack{i=1 \\ i\neq B}}^{N-1} \psi_i(\mathcal{Y}_i)\Bigg) \otimes \psi_B(\mathcal{Y}_B) \otimes \\ \left(\psi_N(\mathcal{Y}_N)^{\downarrow C_N \cap C_B}\right)\Bigg)^{\downarrow C_1}.\end{aligned} \tag{16}$$

In other words, the problem can be converted to an equivalent marginalization over a graph with one less clique in which the potential for C_B has been replaced according to:

$$\psi_B \leftarrow \psi_B \otimes \left(\psi_N^{\downarrow C_N \cap C_B}\right). \tag{17}$$

By repeatedly eliminating cliques in this way we can systematically remove cliques until there is only one remaining, C_1. Any further summation which is required (either to give marginals over a smaller subset of vertices, or to calculate the partition function) can be performed explicitly.

3.2 MESSAGE PASSING

The result of the elimination illustrated in (17) can be interpreted in terms of a message passed from C_N

[1] While our method is applicable to any summation of this form, we will focus on the application to probability distributions in this paper.

to the rest of the graph. Messages are passed between cliques along edges in a *junction tree* [3]. Let $\mu_{i\to j}(\mathcal{Y}_j)$ be the message passed from C_i to C_j. The form of the message is a list of labeled partitions of the intersection $C_i \cap C_j$, each of which has an associated scalar value. The messages are updated iteratively according to the rule:

$$\mu_{i\to j}(\mathcal{Y}_j) \leftarrow \sum_{\mathcal{Y}_i \backsim \mathcal{Y}_j} \psi_i(\mathcal{Y}_i) \prod_{\substack{k\in\mathcal{N}(i)\\ k\neq j}} \mu_{k\to i}(\mathcal{Y}_i), \quad (18)$$

with the outgoing messages from a clique being updated once all incoming messages from the other neighboring cliques $\mathcal{N}(\cdot)$ have been received. As the junction tree has no cycles, this process will terminate after a finite number of iterations. Having updated all of the messages, it is then possible to find f_s using

$$f_s(\mathcal{Y}_1) = \psi_1(\mathcal{Y}_1) \prod_{k\in\mathcal{N}(1)} \mu_{k\to 1}(\mathcal{Y}_1). \quad (19)$$

Having defined the algorithm formally, it is useful to also give an intuitive interpretation. The message passed from C_i to C_j can be interpreted as a statement summarizing the values of the 'upstream' potentials for labeled partitions which are consistent with each labeled partition of the separator between C_i and C_j. See Figure 2 for an example of the message passing process. As is the case with the usual form of the sum-product algorithm, the same messages are used in computing different marginals. Marginal distributions for all cliques can be found simultaneously with a single bidirectional pass of the message update rule.

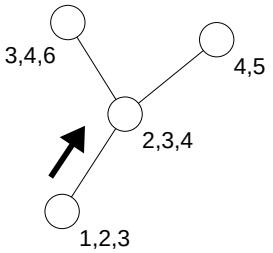
3.3 THE MAX-PRODUCT ALGORITHM

Just as is the case for the usual form of the sum-product algorithm, it is possible to replace the summation in (14) with a maximization to obtain the max-product algorithm. This is equivalent to a redefinition of marginalization to represent the maximum valuation consistent with the sub-partition rather than the sum over all valuations. This algorithm is used to compute maximizations, for example the configuration of C_1 in the most probable labeled partition,

$$\mathcal{Y}_1^{\mathrm{MAX}} = \arg\max_{\mathcal{Y}_1} \max_{\mathcal{Y}\backsim\mathcal{Y}_1} P^*(\mathcal{Y}). \quad (20)$$

In the context of probabilistic inference, this is necessary when searching for the most probable configuration. Message passing is done in the same way as described above, with a modified message update rule.

$$\mu_{i\to j}(\mathcal{Y}_j) \leftarrow \max_{\mathcal{Y}_i \backsim \mathcal{Y}_j} \psi_i(\mathcal{Y}_i) \prod_{\substack{k\in\mathcal{N}(i)\\ k\neq j}} \mu_{k\to i}(\mathcal{Y}_i). \quad (21)$$



(a)

Potential	Value
$\psi_{ij}(t_i = t_j, y_i = y_j)$	0.6
$\psi_{ij}(t_i \neq t_j, y_i = y_j)$	0.4
$\psi_{ij}(t_i \neq t_j, y_i \neq y_j)$	0.2

(b)

Partition	Label	Value
(123)	+	0.216
(123)	-	0.216
(12)(3)	+, +	0.096
(12)(3)	+, -	0.024
⋮	⋮	⋮
(1)(2)(3)	-,-,-	0.064

(c)

Partition	Label	Value
(23)	+	0.336
(23)	-	0.336
⋮	⋮	⋮
(2)(3)	-,-	0.272

(d)

Figure 2: An example of message passing. (a) The junction tree corresponding to $\mathcal{G}$. (b) The potentials, in this case uniform and independent of data for clarity. (c) The clique potential for the clique consisting of vertices 1, 2 and 3. (d) The message passed from (123) to (234), concerning labeled partitions of vertices 2 and 3.

Having updated all of the messages, $\mathcal{Y}_1^{\mathrm{MAX}}$ can be found using

$$\mathcal{Y}_1^{\mathrm{MAX}} = \arg\max_{\mathcal{Y}_1} \psi_1(\mathcal{Y}_1) \prod_{k\in\mathcal{N}(1)} \mu_{k\to 1}(\mathcal{Y}_1). \quad (22)$$

To find the global maximum configuration, we repeat the above for all possible roots, and reconstruct the global partition as the union of the local configurations (which will be consistent with one another).

Again, it is instructive to consider the intuitive meaning of the messages. In this case they can be interpreted as statements about the maximum value that can be achieved 'upstream' as a function of the clique separator configuration. When the next cluster computes its maximum configuration, the contribution of downstream potentials can therefore be incorporated from the messages rather than having to be recomputed from scratch each time.

3.4 EDGE-DUAL REPRESENTATION

Let us consider two alternative representations which cast the inference task so that it can be solved using the standard forms of the sum-product and max-product algorithms. In the first of these techniques, rather than working with partitions, a 'part ID' is assigned to each vertex. The corresponding partition is therefore defined so that contiguous regions with the same part ID are assigned to the same part. To allow

for labeled partitions, a separate set of part IDs must be reserved for each label.

This approach has several problems. Firstly, we must ensure that enough part IDs are available to realize all possible partitions. Depending on the structure of $\mathcal{G}$, a lower bound on the minimum number required is the size of the largest clique. In practice the required number will be greater than this. In general, this means that inference will be significantly slower than the equivalent binary labeling problem.

A more serious drawback of this approach is that it introduces bias into the results; finding the most probable assignment of part IDs is not equivalent to finding the most probable partition; the latter marginalizes over the multiple assignments of IDs which correspond to the same partition.

An alternative representation which avoids these problems is to use indicator variables, $\check{\boldsymbol{x}}(\mathcal{Y})$, for each edge in $\mathcal{G}$. For binary labels, these variables are over a set of six values: two states corresponding to segments belonging to the same part with each label, and four corresponding to different parts with all four combinations of labels. To construct a graphical model for these variables, we define the *edge-dual graph*:

Definition 3. *For any graph $\mathcal{G}$, the **edge-dual graph**, $\check{\mathcal{G}} = \left(\check{\mathcal{V}}, \check{\mathcal{E}}\right)$ contains one vertex for each edge in $\mathcal{G}$. Vertices in $\check{\mathcal{G}}$ are connected by an edge if and only if all vertices connected to their corresponding edges in $\mathcal{G}$ belong to the same clique.*

An example of an edge-dual graph is shown in Figure 3. Every labeled partition of $\mathcal{G}$ corresponds to a unique configuration of the edge-dual vertices, but there are configurations of the edge-dual vertices which do not correspond to labeled partitions. Hence,

Definition 4. *A configuration of the edge-dual vertices is **valid** if and only if it corresponds to a labeled partition of $\mathcal{G}$.*

Invalid configurations arise when pairwise constraints yield contradictory information; following one path between two vertices on $\mathcal{G}$ indicates that they are in the same part, whereas another path indicates that they are not, or their labels disagree. It is possible to establish the validity of a configuration using only calculations local to cliques on $\check{\mathcal{G}}$.

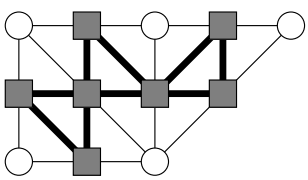


Figure 3: An example of an undirected graph (circular vertices and light lines) and the corresponding edge-dual graph (square vertices and heavy lines).

Suppose $P^*\left(\check{\boldsymbol{x}}\left(\mathcal{Y}\right)\right)$ is a probability distribution over labeled partitions of $\mathcal{G}$ as represented by the edge-dual variables. We are generally interested in operations such as the summation of P^* over all partitions. Rather than expressing the summation in terms of partitions, we can work directly with $\check{\boldsymbol{x}}$, provided that the summation is limited to those configurations which are valid. This can be achieved by introducing an indicator function, $\mathbb{I}\left(\check{\boldsymbol{x}}\right)$, which takes the value 1 if $\check{\boldsymbol{x}}$ is valid and 0 otherwise,

$$\sum_{\mathcal{Y}} P^*\left(\check{\boldsymbol{x}}\left(\mathcal{Y}\right)\right) = \sum_{\check{\boldsymbol{x}}} \mathbb{I}\left(\check{\boldsymbol{x}}\right) \cdot P^*\left(\check{\boldsymbol{x}}\right). \tag{23}$$

There is a one-to-one correspondence between cliques in $\mathcal{G}$ and $\check{\mathcal{G}}$, so functions which factor according to $\mathcal{G}$ also factor according to $\check{\mathcal{G}}$. If P^* factors, we can write

$$\sum_{\mathcal{Y}} P^*\left(\check{\boldsymbol{x}}\left(\mathcal{Y}\right)\right) = \sum_{\check{\boldsymbol{x}}} \left(\prod_i \mathbb{I}_i\left(\check{\boldsymbol{x}}_i\right) \cdot \check{\psi}_i\left(\check{\boldsymbol{x}}_i\right) \right), \tag{24}$$

where i ranges over the cliques of $\check{\mathcal{G}}$. In (24), the local nature of $\mathbb{I}$ has been used to factor it as well as P^*. The result is a sum over a function which factors according to $\check{\mathcal{G}}$, so it can be found using the standard sum-product algorithm.

As there is a one-to-one correspondence between valid edge-dual configurations, and labeled partitions of $\mathcal{G}$, this algorithm is in many respects equivalent to that presented in Section 3.1. However, in two important respects it is less efficient. Firstly, as the sum includes edge-dual configurations which are invalid, the number of terms in the sum is significantly greater. Secondly, it is necessary to determine the validity of the current configuration for each term, which introduces additional overhead. The algorithm presented in Section 3.1 may be regarded as an efficient implementation of this algorithm, where the validity of configurations is precomputed, and only those which are valid are included in the sum.

3.5 COMPLEXITY

The dominant factor in the complexity of the message passing algorithm is the time taken to process all possible partitions of the largest clique. Table 1 lists the number of possible configurations for the various cases. It can be seen from the table that the method described in Section 3 offers a considerable improvement in the complexity of the calculations.

Table 1: Sizes of the message tables for each of the methods. (a) Unlabeled Partitions (these are the Bell numbers). (b) Binary labeled partitions (c) Binary labeled edge-dual representation. (d) Binary labeled part IDs (lower bound).

	Clique Size					
	2	**3**	**4**	**5**	**6**	**n**
(a)	2	5	15	52	203	Bell no. B_n
(b)	*6*	*22*	*94*	*454*	*2430*	A001861 [6]
(c)	6	216	46656	6.0×10^7	4.7×10^{11}	$6^{n(n-1)/2}$
(d)	16	216	4096	1.0×10^5	3.0×10^6	$(2n)^n$

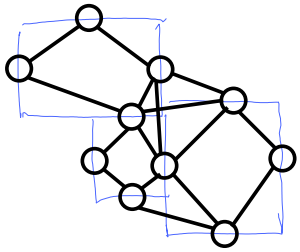


Figure 4: An example of an undirected graph constructed from the input data in which each vertex represents an ink fragment.

4 APPLICATION TO INK DATA

In this section we apply the algorithm developed in Section 3 to the task of parsing hand-drawn ink diagrams, focusing on the particular problem of grouping electronic ink strokes into perceptually salient objects and labeling the resulting groups. We demonstrate our approach on organization charts such as that shown in Figure 5, where objects are labeled as either containers or connectors. However, the method is general and may be applied to a wide variety of related problems.

4.1 PRE-PROCCESSING

The input data is a set of ink strokes, which may span multiple objects. The first stage is to split the strokes into fragments, which are assumed to belong to a single object, by dividing each stroke into sections which are straight to within a given tolerance.

Having fragmented the strokes, we build an undirected graph, $\mathcal{G}$, containing one vertex for each ink fragment (See Figure 4). This is the graph which will be partitioned to obtain the grouping of ink fragments. In our algorithm, $\mathcal{G}$ is constructed by first building a candidate graph (which is not necessarily triangulated) by connecting all pairs of fragments satisfying an appropriate distance constraint. Additional edges are added to create a triangulated graph, and pairwise feature vectors are generated for all edges on the new graph, including those which were added during triangulation. This approach gave a mean tree-width of 4.0 when applied to our training database. By modifying the algorithm to constrain the tree-width, an adjustable compromise between speed and accuracy can be obtained.

Table 2: Labeling errors for the three models. Results are the mean of three cross-validation splits. Relative differences are shown between models L and LI, and between LI and PLI. The mean relative differences are aggregations of the differences for each split, rather than the differences between the means for individual models. This is to reduce the effect of systematic variation between splits.

L	8.5%
LI	4.5%
% Δ LI/L	$-48.9\% \pm 24.9\%$
PLI	2.6%
% Δ PLI/LI	$-42\% \pm 8\%$

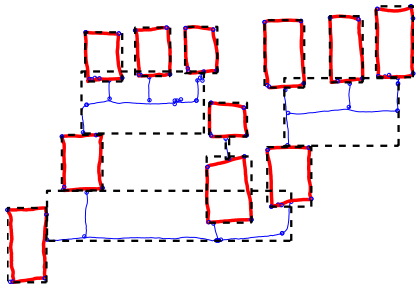


Figure 5: Example labelings and groupings: the most probable partition and labeling using model PLI. Heavy lines indicate fragments which have been classified as containers and lighter lines indicate connectors. Groups of fragments which belong to the same part are outlined using a dashed box. (Image rotated from original.)

4.2 FEATURES AND PRIORS

We chose features to reflect the spatial and temporal distribution of ink strokes, for example lengths and angles of fragments, whether two fragments were drawn with a single stroke, and the temporal ordering of strokes. We also used a number of 'template' features which were designed to capture important higher level aspects of the ink, such as the presence of T-junctions.

We use Gaussian priors, with correlations specified between the priors for weights corresponding to related features. In total 61 unary features and 37 pairwise features were used.

4.3 RESULTS

To test the performance of the method, we used a database of 40 example diagrams, consisting of a total

of 2157 ink fragments. Three random splits were generated, each consisting of 20 examples used for training and 20 used for evaluation. Training was performed by finding the MAP weights as described in Section 2.1. The models were tested by finding the most probable partition and labeling as described in Section 2.2, and counting errors made against ground-truth data.

For comparison, we also consider two related models which model labeling only, without considering partitioning. The first of these models has a similar form to that described in Section 2, but uses pairwise potentials given by

$$\psi_{ij}^{(2)}\left(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}\right) = \begin{cases} \phi\left(\boldsymbol{v}_{\mathrm{s}} \cdot \boldsymbol{f}_{ij}\left(\boldsymbol{x}\right)\right) & \text{if } y_i = y_j \\ \phi\left(\boldsymbol{v}_{\mathrm{d}} \cdot \boldsymbol{f}_{ij}\left(\boldsymbol{x}\right)\right) & \text{if } y_i \neq y_j \end{cases}, \quad (25)$$

where $\boldsymbol{v}_{\mathrm{s}}$ and $\boldsymbol{v}_{\mathrm{d}}$ are weights corresponding to vertices i and j having the same and different labels respectively. The second related model does not use pairwise potentials at all — ink fragments are labeled independently of the other labelings. In the following, we refer to the full model performing labeling and partitioning as model PLI. LI is the model performing labeling only with pairwise potentials, and L is the model with unary potentials only.

Labeling error rates are shown in Table 2. Figure 5 shows the output of the algorithm on an example diagram. Further examples are available online at `http://research.microsoft.com/~szummer/aistats05/`.

5 DISCUSSION

The results given in Section 4.3 show that our approach is capable of providing high-quality labeled partitions. The data also illustrate an important point; simultaneous labeling and partitioning produces a significant improvement in labeling performance. This is easily understandable — the constraint that vertices within the same part must be labeled identically provides strong evidence for the labeling part of the algorithm, and the boundaries between regions of different labels are strong candidates for part boundaries. Hence the two aspects of the algorithm reinforce each other.

There are a number of extensions to the model which have not been discussed in this paper. The most straightforward is the incorporation of other local constraints, such as known labels of particular vertices, or information concerning the relationship of two vertices in the partition. These can easily be included through additional potentials which assign zero probability to configurations violating the constraints, and in the context of the ink parsing provide a valuable method for incorporating user feedback. It seems that more complex information, such as priors over the number of parts, can be incorporated by increasing the amount of information passed in the messages.

In some applications the maximum clique size may be too large for exact inference to be feasible, motivating approximate methods. Monte Carlo techniques have already been applied to problems of this sort [1], but it is desirable to apply alternative approximations such as loopy belief propagation, variational inference or expectation propagation.

6 CONCLUSION

We have presented a probabilistic model over labeled partitions of an undirected graph, and have shown that the structure of the graph may be used to efficiently perform exact inference with message passing algorithms. We have demonstrated the application of the model to the task of parsing hand-drawn diagrams. Our experiments illustrate that it is possible to obtain high-quality results using this technique. The results obtained prove that in our applications, labeling accuracy is improved by performing partitioning at the same time.

Acknowledgements

We would like to thank Thomas Minka, Yuan Qi and Michel Gangnet for helpful advice and discussion, and for providing excellent software that allowed the work presented in this paper to be completed. We are also grateful to Hannah Pepper for collecting our ink database.

References

[1] A. Barbu and S. Zhu. Graph partition by Swendsen-Wang cuts. In *ICCV*, 2003.

[2] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[3] S. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[4] X. Liu and D. Wang. Perceptual organization based on temporal dynamics. In *NIPS*, volume 12, 2000.

[5] P. Shenoy and G. Shafer. Axioms for probability and belief-function propagation. In *Readings in uncertain reasoning*, Morgan Kaufmann, pages 575–610, 1990.

[6] N. Sloane. The On-Line Encyclopedia of Integer Sequences, 2004. `http://www.research.att.com/projects/OEIS?Anum=A001861`

[7] C. Sutton, K. Rohanimanesh, and A. McCallum. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *ICML*, 2004.

Restructuring Dynamic Causal Systems in Equilibrium

Denver Dash
Intel Research
3600 Juliette Lane, SC12-303,
Santa Clara, CA 95054, USA
denver.h.dash@intel.com

Abstract

In this paper I consider general obstacles to the recovery of a causal system from its probability distribution. I argue that most of the well-known problems with this task belong in the class of what I call *degenerate* causal systems. I then consider the task of discovering causality of dynamic systems that have passed through one or more equilibrium points, and show that these systems present a challenge to causal discovery that is fundamentally different from degeneracy. To make this comparison, I consider two operators that are used to transform causal models. The first is the well-known *Do* operator for modeling manipulation, and the second is the *Equilibration* operator for modeling a dynamic system that has achieved equilibrium. I consider a set of questions regarding the commutability of these operators i.e., whether or not an equilibrated-manipulated model is necessarily equal to the corresponding manipulated-equilibrated model, and I explore the implications of that commutability on the practice of causal discovery. I provide empirical results showing that (a) these two operators sometimes, but not always, commute, and (b) the manipulated-equilibrated model is the correct one under a common interpretation of manipulation on dynamic systems. I argue that these results have strong implications for causal discovery from equilibrium data.

1 Introduction

Causal Discovery refers to a special class of statistical analysis that seeks to infer, from a set of data, information about causal relations between variables. There has been much success on the topic of causal discovery in the past decade in Artificial Intelligence [Spirtes *et al.*, 2000; Verma and Pearl, 1991; Heckerman *et al.*, 1999; Tian and Pearl, 2001], building on structural-equation modelling techniques originating in early econometrics [cf., Simon, 1953; Wold, 1954].

There are, as one might expect, many difficulties with inferring reliable causal relationships from data. Latent common causes confounding relations between the observed variables, nonlinearity, acyclicity, and violations of faithfulness due to the cancelling of multiple causal paths are just a few. Identifying prospective pitfalls such as these is the critical first step to developing techniques to handle them in a principled way.

This paper exposes another obstacle to causal discovery that is likely prevalent and important, but is not currently being addressed by causal discovery research. I describe this event as *a violation of equilibration-manipulation commutability* (or *EMC violation*, for short), for reasons that I hope to make clear shortly. I show that EMC violation occurs in static systems, but when those systems have an underlying dynamics which have passed through some equilibrium points. I illustrate the existence of EMC violation by example. Then as further validation, I provide empirical results showing that EMC violation occurs in practice, and its occurrence depends on the time-scale at which the data is being collected relative to the important time-scales of the underlying dynamic systems. I argue that, since many real-world static systems are essentially equilibrium points of underlying dynamic systems, EMC violation is likely to be a common occurrence. I also argue that one can reduce the chance of an EMC violation when building causal models by taking care when choosing the set of variables to include in one's model.

In Section 2, I define some background concepts and explore known obstacles to causal discovery; in Section 3, I show a motivating example of a dynamic causal system going through equilibrium, I define the

EMC property and show why it is important; in Section 4 I show empirically how an EMC violating system can impact causal discovery in practice; in Section 5 I sketch two theorems that show sufficient conditions for systems to violate and obey EMC, and finally I conclude in Section 6.

2 Background Concepts

In this section I define a *causal system*, I explore known obstacles to causal discovery, I introduce the EMC questions and I demonstrate why these questions are important.

2.1 Causal Discovery

I define a *causal system* [c.f., Pearl, 2000] in terms of a set of structural equations:

Definition 1 (causal system) *A causal system over a set of variables* $\mathbf{V}$ *is a 4-tuple* $\langle \mathbf{U}, \mathbf{V}, \mathbf{E}, \phi \rangle$*, where* $\mathbf{U}$ *is a set of random variables that are determined outside the system ("exogenous variables"),* $\mathbf{V} = \{V_1, V_2, \ldots V_n\}$ *is a set of n variables determined by the system ("endogenous variables"),* $\mathbf{E}$ *is a set of n equations, and* $\phi : \mathbf{V} \rightarrow \mathbf{E}$ *is an onto mapping such that for every* $V_i \in \mathbf{V}$*,* $\phi(V_i)$ *can be written as* $V_i = f_i(\mathbf{Pa}_i, \mathbf{U}')$*, where* $\mathbf{Pa}_i \subseteq \mathbf{V} \setminus \{V_i\}$*,* $\mathbf{U}' \subseteq \mathbf{U}$*, and* f_i *is a function.*

A causal system defines a directed graph over variables in $\mathbf{V}$ as follows: For each V_i, let $\phi(V_i)$ be written as $V_i = f_i(\mathbf{Pa}_i, \mathbf{U}')$, and draw an arc from all variables $P_i^j \in \mathbf{Pa}_i \cup \mathbf{U}'$ to V_i. A graph constructed in this way is called a *causal graph*, and if P_i^j is a parent of V_i in this graph then P_i^j is a *cause* of V_i, and V_i is an *effect* of P_i^j. All Bayesian networks can be mapped onto a causal system [Druzdzel and Simon, 1993], but the converse is not true, e.g., causal systems can define cyclic graphs.

All randomness in a causal system is induced by the exogenous variables, which are assumed to be controlled by external forces and therefore are treated as random variables. To say that an equation $\phi(V_i)$ is *deterministic* means that $\mathbf{U}' = \emptyset$, in which case, V_i is a deterministic function of $\mathbf{Pa}_i$. Because the variables in $\mathbf{U}$ are random variables, and because in general the variables in $\mathbf{V}$ depend on $\mathbf{U}$, the causal system $S = \langle \mathbf{U}, \mathbf{V}, \mathbf{E}, \phi \rangle$ will define a probability distribution P over the set $\mathbf{V}$. A common assumption is to assume that each endogenous variable V_i in a causal system S depends on a single exogenous variable U_i and for all i, j, U_i is independent of U_j.

Causal systems such that all f_i functions are linear and all $U_i \in \mathbf{U}$ are normally distributed are called *linear structural equation models*, and for decades these have been widely used in econometrics and the social sciences to model causality [c.f., Simon, 1953; Wold, 1954].

Causal discovery or *causal inference* is the task of analyzing a probability distribution P, and possibly other background information I, to reconstruct the causal system S that generated P. In practice, however, even if P is known exactly, causal inference can do no better than identifying the *set* of causal models that define distributions identical to P that are consistent with I. Probability distributions which do not uniquely define a causal system are the most commonly observed obstacles to causal discovery. I call a causal system for which that is the case *degenerate*. Specific instances of features of causal systems that lead to degenerate probability distributions have been identified and are discussed in the next section.

2.2 Degenerate Causal Systems

Examining the conditional independence relations present in the probability distribution is a key method for causal discovery. Obviously, it is the presence of these relations that increases the specificity of the distribution and makes identification of causal relations possible. One of the most general problems one encounters when trying to perform causal inference from independence relations is a lack of *faithfulness*:

Definition 2 (faithfulness) *A probability distribution* $P(\mathbf{V})$ *over a set of variables* $\mathbf{V}$ *is* faithful *to a directed graph* G *over* $\mathbf{V}$ *if, for every conditional independence relation* $(V_1 \perp V_2 \mid \mathbf{V}')$ *in* P*, there exists a d-separation condition* $(V_1 \perp_d V_2 \mid \mathbf{V}')$ *in* G *and vice-versa*[1]*, for* V_1*,* $V_2 \in \mathbf{V}$ *and* $\mathbf{V}' \subset \mathbf{V}$*.*

If P is faithful to G then G is called a *perfect map* or *p-map* of P. P is called *causally faithful* to G when P is faithful to G, and G is a causal graph. Specific cases in which unfaithful distributions can be generated from real causal systems have been identified in Spirtes *et al.* [2000]. Two in particular are:

- **Determinism:** when a variable in a causal system depends deterministically on other variables. For example, in the causal graph with three variables $\{A, B, C\}$ such that: $A \rightarrow B \rightarrow C$ and $A \rightarrow C$, if C is a deterministic function of A, then C is independent of B given A although that d-separation condition does not exist in the causal graph.

[1] Some definitions of faithfulness do not require the converse.

- **Cancelling causal paths:** when two or more causal paths exactly cancel out. This event can make two or more variables non-correlated although they are causally connected. Although this is possible in principle, Spirtes *et al.* [2000] argue that its occurrence has Lebesque measure zero.

Other reasons for causal degeneracy are:

- **Statistical Indistinguishability**: when there exist other causal structures that have the same set of *adjacencies* and *v-structures*.
- **Lack of causal sufficiency.** A common cause $C \leftarrow A \rightarrow B$ will cause a dependence between C and B in the probability distribution over these variables. If A has been marginalized out of the distribution P, it becomes difficult to decide whether there is a direct causal arc between C and B given only P.
- **cyclic causality:** when a directed cycle exists in the causal graph. Although the physical relevance of these systems can be argued, they are not forbidden by definition, and the implications of their existence on independence relations is not fully explored.

One mitigating fact for all of these obstacles is that their occurrences are all detectable post-causal-discovery, at least sometimes: Determinism and cancelling causal paths will be detectable in the parameters of the model; statistical indistinguishability will be identifiable from the structure of the model; hidden common causes and cyclic causality can sometimes be detected: for example, when their presence causes many v-structures, the PC algorithm for causal discovery [Spirtes *et al.*, 2000] can produce bi-directed arcs or cycles, respectively.

Degenerate causal systems are on one hand problematic, but on the other hand are easy to understand. In the next section I introduce a qualitatively different type of obstacle to causal discovery which, in the author's opinion, is much less transparent and therefore more interesting than causal degeneracy. I call it "violation of *Equilibration-Manipulation Commutability*."

3 The EMC Property

When a causal system is based on a set of differential (or difference) equations, the probability distribution it specifies will not be static, but instead will be a function of time. The evolution of the probability distribution should be predictable. For example, consider the following discrete-time first-order difference system where the change, ΔX, in some variable X, is determined by a linear balance of factors:

$$
\begin{aligned}
X^0 &= x_0 && (1)\\
F_1^t &= \alpha_1 U_1^0 && (2)\\
F_2^t &= \alpha_2 X^t + U_2^0 && (3)\\
\Delta X^t &= \alpha_3 F_1^t + \alpha_4 F_2^t + U_x^0 && (4)\\
X^{t+1} &= X^t + \Delta X^t, && (5)
\end{aligned}
$$

where all α_i are constants. In this system, I have assumed that the exogenous variables $\{U_1^0, U_2^0, U_x^0\}$ are static throughout time (which is why they have a fixed $t = 0$ superscript). The causal graph for this system unrolled out to three time slices is shown in Figure 1-(a). The dotted boxes around F_1, F_2 and ΔX are used to denote the fact that the exogenous variables are static through time and are thus parents of those variables in each time slice. For conciseness I will use a shorthand graph, based on the notation of Iwasaki and Simon [1994], where arcs that occur through time are shown with dotted lines, instantaneous arcs are shown in solid, and static exogenous arcs shown dashed. The corresponding shorthand graph for our toy example is shown in Figure 1-(b). It should be emphasized that, although the shorthand version of this graph contains a directed cycle, it represents an acyclic dynamic graph. Sometimes I may also drop the exogenous variables

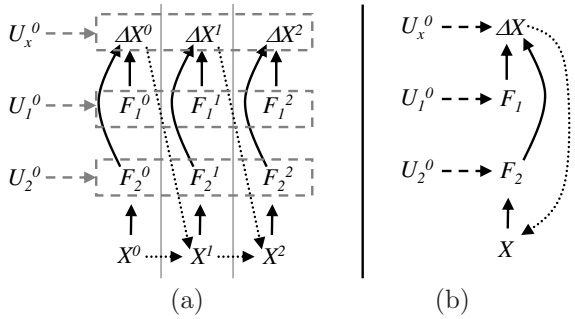


Figure 1: (a) A toy example dynamic causal graph based on difference equations and with static exogenous variables, and (b) the same graph in "shorthand" form.

from this graph to emphasize the endogenous causal relations.

If one considers what the probability distribution over the endogenous variables of this system at the nth time slice will look like, one needs only to expand this system out n slices and marginalize out all variables from previous time-slices to see the causal structure and to generate the probability distribution of the variables at the nth slice. The problem, however, is, if one waits long enough, it may very well be that this system achieves *equilibrium*, at which time there will

be a qualitative change in the probability distribution. Specifically, at equilibrium, $\Delta X^t = 0$, so the system of equations reduces to:

$$F_1 = \alpha_1 U_1^0 \quad (6)$$
$$F_2 = -\alpha_3 F_1/\alpha_4 - U_x^0 \quad (7)$$
$$X = F_2/\alpha_2 - U_2^0 \quad (8)$$

In the distribution defined by this system, although U_2^0 is a *parent* of F_2 in the original causal system, it is marginally independent of F_2 in the equilibrium equation system. This fact is obvious by looking at the independence graph of this system shown in Figure 2, and it can also be easily derived from the equation system assuming independent exogenous variables.

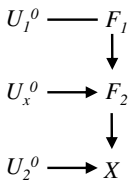


Figure 2: The independence graph of the equilibrium distribution defined by our toy causal system. Although U_2^0 is a parent of F_2 in the original dynamic system, it is marginally independent of F_2 in the equilibrium distribution.

This example illustrates a novel unanswered question associated with dynamic causal systems. Namely, if a causal system is a set of equations with some structure imposed upon it, and if, when a non-structural equation system goes through equilibrium, the equations go through a qualitative change, how should the causality of a system passing through an equilibrium point be modelled? That is, if some equations and variables are dropping out, how should the remaining equations be structured? In the above example, Equation 4 was originally used to determine ΔX; however in equilibrium ΔX has dropped out and Equation 4 has changed into Equation 7 and now "determines" F_2. In fact, if one were to learn a causal graph given the equilibrium distribution, obviously one would in general recover a totally different causal structure than would be recovered from the non-equilibrium system at some arbitrary time slice n. I show this fact empirically in Section 4.

It has been argued by Iwasaki and Simon [1994] that the causal relations governing a dynamic system can change as the time-scale of observation of the system is increased. In particular, they introduce the *Equilibration* operator that they argue produces the causal relations of a system in equilibrium given the dynamic (non-equilibrium) causal system. A detailed treatment of the equilibration operator is beyond the scope of this paper (see Iwasaki and Simon [1994] for more details), a sketch of the operator is as follows:

Definition 3 (Equilibration (sketch)) *Let $M = \langle \mathbf{U}, \mathbf{V}, \mathbf{E}, \phi \rangle$ be a causal model, and let $X \in \mathbf{V}$ be a dynamic variable in M. Equil(M, X) is a causal model $M' = \langle \mathbf{U}', \mathbf{V}', \mathbf{E}', \phi' \rangle$ that is defined by:*

1. *$\mathbf{V}'$ is equal to $\mathbf{V}$ with all of X's derivatives removed.*
2. *$\mathbf{E}'$ is equal to $\mathbf{E}$ with all integration equations removed, and*
3. *$\phi' : \mathbf{V}' \rightarrow \mathbf{E}'$ is an onto mapping.*

In general, such an operation may not define a unique mapping ϕ'; however, in the remainder of the paper I assume that ϕ' is unique and only present examples for which that is the case.

As we have done with our toy example, the equilibration operator formally makes the assumption of equilibrium which causes a modification to the equations of the system, then it recovers a structure consistent with the remaining set of equations. In many cases, there is a unique (independence) structure remaining (as in Figure 2). Iwasaki and Simon argue that under these circumstances that structure must be the causal structure of the system under equilibrium.

The *Do* operator, $Do(M, \mathbf{U} = \mathbf{u})$, is another operator of a causal system that transforms a causal model M to a new causal model M' where a subset of variables $\mathbf{U}$ in M' are fixed to specific values independent of the causes of $\mathbf{U}$. On the other hand, the *Equilibration* operator, *Equil*(M, X), transforms the model M with a dynamic (time-varying) variable X to a new causal model M' where X is static. This paper considers the relationship between these two operators. In particular I am interested in the what I call the *Equilibration Manipulation Commutability* property, or the *EMC* property for short:

Definition 4 (EMC Property) *Let $M(\mathbf{V})$ be a causal model over variables $\mathbf{V}$. M satisfies the Equilibration-Manipulation Commutability (EMC) property iff*

$$Equil(Do(M, \mathbf{U} = \mathbf{u}), X) = Do(Equil(M, X), \mathbf{U} = \mathbf{u}),$$

for all $\mathbf{U} \subseteq \mathbf{V}$ and all $X \in \mathbf{V}$.

In this paper, I consider the following set of questions (hereafter referred to as *the EMC questions*):

1. Does the EMC property hold for all dynamic causal models?

2. Does the EMC property hold for any dynamic causal models?

3. Under what conditions is the EMC property guaranteed to hold?

4. Under what conditions is the EMC property guaranteed to be violated?

5. In general, is it sensible to reason about causality in a dynamic system that has passed through some equilibrium points?

These questions are important for at least the following reason: Very often in practice a causal model is first built from either data or knowledge of equilibrium relationships, and then causal reasoning is performed on that model. This common approach takes path A in Figure 3. When a manipulation is performed on a

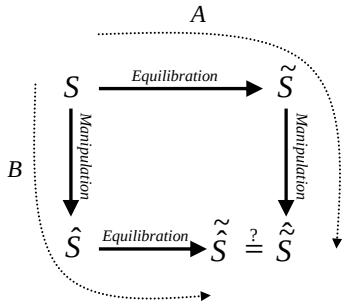


Figure 3: The EMC Questions consider under what conditions the *Do* operator commutes with the Equilibration operator operating on a dynamic causal model S.

system, however, the state of the system in general becomes "shocked" taking the system out of equilibrium, a situation which is modelled by path B in Figure 3. The validity of the common approach of taking path A thus hinges on the answers to the EMC Questions.

This paper primarily answers EMC Questions 1, 2, and 5. The toy example I prove by example and by empirical tests that the answer to Question 1 is "No" and that of Question 2 is "Yes". These results in turn implies that the answer to Question 5 is "Sometimes". The answers to Questions 3 and 4 are addressed in Dash [2003], but I will sketch those results here as well.

4 Discovery from Data: Empirical Results

The previous section presented an example which implied that the answer to EMC Question 1 is "no". This section addresses the EMC Questions using empirical studies. I performed numerical simulations of a dynamic system to demonstrate that as the time scale was increased enough so that an equilibration could occur, the causal structure that was learned from data corresponds to the structure obtained by applying the *Equilibration* operator to the dynamic model. This fact is significant because it indicates that whenever a causal structure that is learned from equilibrium data is used for causal reasoning, then Path A of Figure 3 is being taken: if the EMC property does not hold for the model being used then subsequent causal reasoning will produce incorrect results.

Consider the causal system of five variables $\{Q_{in}, Q_{out}, D, K, P\}$ defined by Equations 9–13 below.

$$K = K_0 \quad (9)$$
$$Q_{in} = Q_0 \quad (10)$$
$$\dot{P} = \alpha_2(\alpha_4 D - P) \quad (11)$$
$$\dot{Q}_{out} = \alpha_3(\alpha_1 KP - Q_{out}) \quad (12)$$
$$\dot{D} = \alpha_0(Q_{in} - Q_{out}) \quad (13)$$

where $\dot{Q}_{out}$, $\dot{D}$ and $\dot{P}$ are the first time-derivatives of Q_{out}, D and P, respectively, and $\alpha_i : i \in \{0, 1, \ldots, 4\}$ are constants.

This system was taken from Iwasaki and Simon [1994]. To give some physical intuition, it roughly approximates a filling-bathtub where water is entering the bathtub from the faucet at a rate Q_{in} liters per second and is exiting the drain at a rate Q_{out} liters per second. The pressure of the water at the base of the drain is P, the depth of the water is D, and the diameter of the drain is K. In this system Q_{in} and K are exogenous and the remaining variables are dynamic. The causal graph of this system is shown in Figure 4.

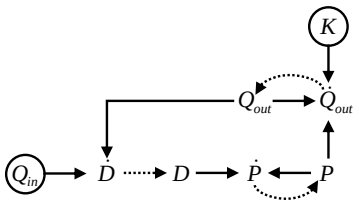


Figure 4: The dynamic causal graph S_0 of the bathtub system.

This system has three dynamic variables, and therefore three possible equilibrations, corresponding to the occurrence of $\dot{P} = 0$, $\dot{Q}_{out} = 0$, and $\dot{D} = 0$. When these conditions occur, Equations 11, 12, and 13 reduce to the equilibrium relations given by Equations 14–16,

respectively:

$$P = \rho g D \quad (14)$$
$$Q_{out} = \alpha_1 K P \quad (15)$$
$$Q_{in} = Q_{out} \quad (16)$$

This system has many potential equilibrium causal orderings depending on the relative time scales of the three variables P, Q_{out} and D, and depending on the time scale at which the system is observed. If, for example, P and Q_{out} both reach equilibrium much sooner than D, and the system is observed before D has reached equilibrium but after P and Q_{out}, then Equations 11 and 12 get replaced by Equations 14 and 15, respectively. The causal ordering of this system is shown in Figure 5.

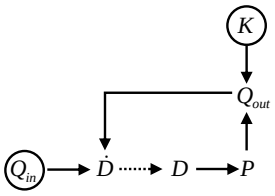


Figure 5: The causal ordering of the bathtub system when P and Q_{out} have been equilibrated but not D.

Because the system of Figure 4 involves three dynamic variables, there exist three important time-scales for this system, controlled by the inverse of the coefficients: $\tau_D \propto 1/\alpha_0$, $\tau_P \propto 1/\alpha_2$, and $\tau_Q \propto 1/\alpha_3$, for D, P and Q_{out} respectively. If $\tau_P \ll \tau_Q \ll \tau_D$, then there will exist four possible equilibrium causal structures learned from data depending on the time, τ, at which the data was observed. These four structures (over variables $\mathbf{V} = \{Q_{in}, Q_{out}, P, T, K\}$) are shown in Figure 6. At $\tau = 0$ each of the five variables in

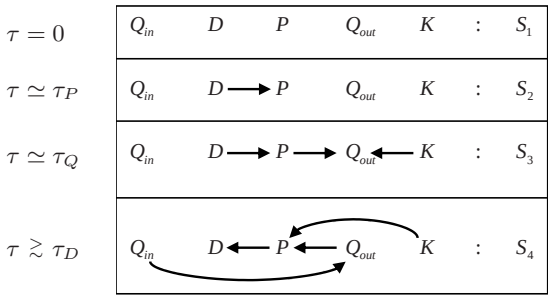


Figure 6: The bathtub system has four correct equilibrium structures depending on the time scale at which the system is observed.

$\mathbf{V}$ are given by their initial conditions and so are exogenous; in this case S_1 will be the structure learned from data. After enough time has passed for P to equilibrate ($\tau \simeq \tau_P$), then Equation 11 reduces to Equation 14, and the structure S_2 will result. After $\tau \simeq \tau_Q$, enough time has passed for Q_{out} to equilibrate, and Equation 12 reduces to Equation 15, resulting in the structure S_3. Finally, after $\tau > \tau_D$, enough time has passed for D to equilibrate and Equation 13 reduces to Equation 16, leading to a drastic restructuring of equations and resulting in model S_4.

I simulated learning over several time-scales for the filling-bathtub system. The following values for constants were used: $\alpha_1 = 1$, $\alpha_0 = 0.005$, $\alpha_2 = 0.05$, and $\alpha_3 = 0.01$. All variables were initialized from the uniform distribution over the interval $(0, 1)$. Independent Gaussian error terms with mean 0 were added to each derivative variable. The error terms for $\dot{D}$ and $\dot{Q}_{out}$ had standard deviation equal to 0.01, and $\dot{P}$ had standard deviation equal to 0.5. I assumed the bathtub was infinitely high (no bound on D was enforced), so given these constants, an equilibrium was guaranteed to exist.

A database of $N = 10000$ records was generated for each of the 29 time-scales given in the set $\mathcal{T} = \{0-10, 20, 30, 40, 50, 80, 100, 125, 150, 200, 250, 300, 500, 750, 1000, 1250, 1500, 1750, 2000\}$, and for each of these databases the PC algorithm was run to retrieve a causal graph. A modified version of PC was used which forbade cycles or bi-directional arrows and randomized the order in which independencies were checked [Dash and Druzdzel, 1999]. Data for each variable took on a continuous range of values, and in all cases the Fisher's-z statistic was used to test for conditional independence using a significance level of $\alpha = 0.05$.

I restricted structure learning to the variables $\{D, P, Q_{out}, Q_{in}$ and $K\}$, namely the variables relevant to the static analysis of this system. This was performed 50 times for each time scale, and the number of times the pattern corresponding to the graphs in Figure 6 were exactly recovered was counted.

The normalized results, showing the empirical probability of retrieving the four structures as a function of the time scale, are shown in Figure 7. For example, when the system is observed just one time step away from the initial conditions, Figure 7 shows that structure S_2 was learned around 45% of the time, structure S_1 was recovered around 6% of the time, structure S_3 was discovered less than 5% of the time, and some other structure was learned the remainder (about 44%) of the time. The time-scales $20 \leq \tau \leq 750$ are excluded from this figure—they produced empirical probabilities of 0 for all four structures. These results show convincingly that as the time-scale increases, the learned causal structure changes in the order predicted by the

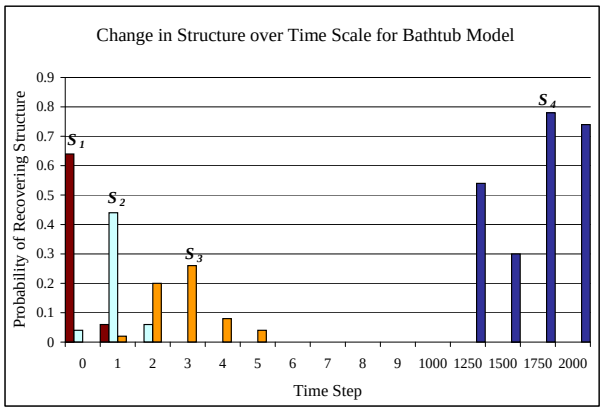


Figure 7: As the time step is varied, each of the four equilibrium structures can be recovered in sequence.

equilibration operator.

It is easy to verify that the EMC property holds when only P or Q_{out} are equilibrated and any of these five variables are manipulated (by verifying that manipulating any variable in S_1 or S_2 results in the same graph as manipulating them in Figure 4 then equilibrating). On the contrary, it is easy to verify that when D is equilibrated, the EMC property is violated: $Do(Equil(S_0, D), D)$ corresponds to S_4 with the arc from P to D removed; whereas $Equil(Do(S_0, D), D)$ (constructed by applying the Do operator to S_0 and then equilibrating all remaining dynamic variables) corresponds to S_3.

5 Theoretical Results: EMC Questions 3 and 4

The results from Section 4 show that in some cases the EMC property is preserved, while in others it is not. While it is beyond the scope of this paper to address the precise conditions when EMC will or will not be violated, I will briefly sketch in this section two results from Dash [2003] with proofs omitted. The first states conditions for which EMC is guaranteed to be violated, the second states conditions for which EMC is guaranteed to be satisfied.

These results involve the concept of a *feedback set*. A feedback set of a variable X in a causal model is the set of variables that are both ancestors and descendants of X in the shorthand causal graph. For example, in Figure 4, the feedback variables of D are $\dot{P}$, P, $\dot{Q}_{out}$, Q_{out} and $\dot{D}$. I let $\boldsymbol{Fb}(X)_M$ denote the set of feedback variables of X in model M.

For the following two theorems, we consider a dynamic causal model $M = \langle \mathbf{U}, \mathbf{V}, \mathbf{E}, \phi \rangle$ and let $M_{\tilde{v}} = \langle \mathbf{U}', \mathbf{V}', \mathbf{E}', \phi' \rangle$ denote the graph that results when a variable $V \in \mathbf{V}$ is equilibrated in M: $M_{\tilde{v}} = Equil(M, V)$. I assume that $M_{\tilde{v}}$ is unique.

Theorem 1 (EMC violation) *If both M and $M_{\tilde{v}}$ are recursive (have acyclic graphs) and there exists any $F \in \boldsymbol{Fb}(V)_M$ such that $F \in \mathbf{V}'$ then $Do(M_{\tilde{v}}, Y) \neq Equil(Do(M, Y), V)$ for any $Y \in \mathbf{V}$.*

For example, in Figure 6, the graph that results when D is equilibrated contains variables that are in the feedback set of D, so the bathtub system violates EMC when D is equilibrated.

Theorem 2 (EMC obeyance) *Let $\Delta^n V$ be the highest derivative (difference) of V in $\mathbf{V}$. If both M and $M_{\tilde{v}}$ are recursive (have acyclic graphs) and $V \in \mathbf{Pa}(\Delta^n V)$, then $Do(M_{\tilde{v}}, Y) = Equil(Do(M, Y), V)$.*

For example, in Figure 4, $\mathbf{Pa}(\dot{P}) = \{P\}$ and $\mathbf{Pa}(\dot{Q}_{out}) = \{Q_{out}\}$, so the bathtub system will obey EMC when either of these variables are equilibrated, as seen in Figure 6.

6 Conclusions

The results of this paper have important consequences for causal discovery. In particular, they emphasize the importance of considering the time-scale of the data being used for causal discovery. If data is recorded of a system for which some variables have achieved equilibrium, then learning a causal graph and using it to predict the effects of manipulating variables in the model amounts to taking path A in Figure 3; however, path B is the correct one to take: if the EMC property does not hold for that model, then incorrect inferences could result.

The fact that taking path A in Figure 3 produces predictions that differ from path B requires us, if we intend to perform causal reasoning with our model, to either ensure that we are taking path B or ensure that we are dealing with models that obey the EMC property. Currently, most work regarding the discovery or building of causal models takes path A and pays no regard to the EMC property. I hope that this work will bring attention to this fact and help to rectify it.

It is a valid question to ask why the EMC property is useful at all. That is, why treat an equation system that has passed through equilibrium as causal? The answer to that question lies in the extreme difficulty of knowing what the important time-scale of an unknown causal system might be. On top of that, to break a system down to its finest time-scale often involves modeling the system in intractable detail. For example, if it were necessary to model the microstates

of a statistical ensemble of particles rather than using the macroscopic laws directly, then modeling the causality of any such system would be impossible.

The problem of identifying the relevant time-scales of a system is especially acute for the task of causal discovery (as opposed to building causal models from expert knowledge), because obviously, if one is trying to learn causal relations from data, it is likely that one is not privy to the details of the underlying dynamics of the system. The positive conclusion of this work is that, for systems that obey EMC, one does not need to consider the system on the shortest possible time scale for the resulting model to accurately reflect causality. The negative conclusion, however, is that at least some knowledge of temporal behavior of the system is likely necessary to ensure that the EMC condition is satisfied, and what knowledge is necessary and sufficient is not yet known.

Although this work raises important objections to some uses of causal reasoning with models learned from data, I believe that the great potential of causal modeling and causal discovery in artificial intelligence make it all the more important for these questions to be explored further and answered as forcefully as possible. The fact that equilibrium causal models are problematic for causal inference should not deter us from developing further the theory that can allow us to build and use them in practice.

7 Acknowledgements

This work was extracted from my PhD thesis [Dash, 2003] pursued at the Intelligent System Program at the University of Pittsburgh in partial collaboration with my advisor Marek Druzdzel. I would like to thank Richard Scheines, Nanny Wermuth and Gregory Cooper for their encouragement for this work.

References

Denver H. Dash and Marek J. Druzdzel. A hybrid anytime algorithm for the construction of causal models from sparse data. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–99)*, pages 142–149, San Francisco, CA, 1999. Morgan Kaufmann Publishers, Inc.

Denver Dash. *Caveats for Causal Reasoning*. PhD thesis, Intelligent Systems Program, University of Pittsburgh, Pittsburgh, PA, April 2003. http://etd.library.pitt.edu/ETD/available/etd-05072003-102145/.

Marek J. Druzdzel and Herbert A. Simon. Causality in Bayesian belief networks. In *Proceedings of the Ninth Annual Conference on Uncertainty in Artificial Intelligence (UAI–93)*, pages 3–11, San Francisco, CA, 1993. Morgan Kaufmann Publishers.

David Heckerman, Christopher Meek, and Gregory F. Cooper. A bayesian approach to causal discovery. In Clark Glymour and Gregory F. Cooper, editors, *Computation, Causation, and Discovery*, chapter four, pages 141–165. AAAI Press, Menlo Park, CA, 1999.

Yumi Iwasaki and Herbert A. Simon. Causality and model abstraction. *Artificial Intelligence*, 67(1):143–194, May 1994.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, Cambridge, UK, 2000.

Herbert A. Simon. Causal ordering and identifiability. In William C. Hood and Tjalling C. Koopmans, editors, *Studies in Econometric Method. Cowles Commission for Research in Economics. Monograph No. 14*, chapter III, pages 49–74. John Wiley & Sons, Inc., New York, NY, 1953.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. Springer Verlag, New York, 1993.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. The MIT Press, Cambridge, MA, second edition, 2000.

Jin Tian and Judea Pearl. Causal discovery from changes. In *Uncertainty in Artificial Intelligence: Proceedings of the Seventeenth Conference (UAI-2001)*, pages 512–521, San Francisco, CA, 2001. Morgan Kaufmann Publishers.

T.S. Verma and Judea Pearl. Equivalence and synthesis of causal models. In P.P. Bonissone, M. Henrion, L.N. Kanal, and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence 6*, pages 255 –269. Elsevier Science Publishing Company, Inc., New York, N. Y., 1991.

Herman Wold. Causality and econometrics. *Econometrica*, 22(2):162–177, April 1954.

Probability and Statistics in the Law

Philip Dawid
Department of Statistical Science
University College London

Abstract

The field of legal reasoning is full of logical subtleties and probabilistic pitfalls. I survey a number of these, pointing out some of the problems and ambiguities, and various attempts to deal with them. Some celebrated court cases are used for illustration.

1 INTRODUCTION

Although the disciplines of Statistics and Law might seem far apart, they share some fundamental interests — in particular, the interpretation of evidence, testing of hypotheses, and decision-making under uncertainty. However, their differing backgrounds and approaches can often lead to misunderstandings, such as in the celebrated Collins case (Fairley and Mosteller 1977). The "New Evidence Scholarship" of the 1970's generated interest within some legal circles in the use of probability as an aid to rational interpretation of evidence, as well as some criticism. Some particularly interesting dialogues were stimulated by the pro-probability paper by Finkelstein and Fairley (1970) (see Tribe (1971b); Finkelstein and Fairley (1971); Tribe (1971a); Brilmayer and Kornhauser (1978); Kaye (1979)); and by the sceptical book by Cohen (1977) (see Schum (1979); Williams (1979); Cohen (1980); Williams (1980); Eggleston (1980)).

In recent years it has become apparent that problems arising in legal settings raise some fascinating and delicate issues of statistical logic, and that, in turn, proper application of statistical reasoning has a rôle to play in the pursuit of justice. In this paper I explore some of these logical issues, with reference to some real cases: see Dawid (2002) for some further background.

2 SUDDEN INFANT DEATHS

There have been a number of recent cases in the UK where two or more young children in a family have died suddenly from no obvious cause, and, even though there is no specifically incriminating evidence, their mother has been convicted of murdering them. In the case of Sally Clark, a paediatrician testified at trial that the probability P that her two sons would have died of SIDS (unexplained natural causes) was 1 in 73 million. That figure was widely and properly criticised, but it can not be denied that P is extremely small. The question is: What are we to make of such "statistical evidence"?

2.1 THE PROSECUTOR'S FALLACY

The correct interpretation $P = \Pr(\mathcal{E} \mid \overline{G})$ (where $\mathcal{E}$ denotes the evidence — here the fact of two infant deaths — G denotes "guilt" and $\overline{G}$ "innocence") is easily distorted into: $P = \Pr(\overline{G} \mid \mathcal{E})$. After all, to say that there is 1 chance in 73 million that the children died of natural causes appears to be just the same as saying that this is the probability that the mother did not kill them — seemingly overwhelming evidence for her being guilty. This mistaken "transposition of the conditional" is so common in court, where it usually favours the prosecution, that it has been termed "the prosecutor's fallacy" (see Gigerenzer (2002) for a clear account of the prosecutor's fallacy, and suggestions as to how it might be avoided). It would have been hard for Sally Clark's jury to ignore this seemingly powerful argument, and they did in fact convict.

2.2 COUNTER-ARGUMENT

There is an obvious counter-argument in this case, which I presented at appeal. We are comparing two alternative hypotheses: two deaths by SIDS, and two deaths by murder. If the chance of the former is relevant, should not that of the latter be equally relevant?

Using UK data, one could argue for a double murder figure of around 1 in 2 billion, to set against the SIDS figure of 1 in 73 million. One can see prosecution and defence brandishing their respective figures in adversarial combat, but the correct approach is to realise that it is their relative, not absolute, values that matter. In fact, their ratio (1/2 billion)/(1/73 million) = 0.0365 can be interpreted as the odds on guilt given the evidence of the two deaths, implying a guilt probability of only 3.5%.

In the event, although the appeal court accepted that there had been some problems with the presentation of the statistical evidence at trial, it was not interested in properly identifying and understanding the logical issues involved. Sally Clark was eventually cleared on entirely unrelated grounds.

3 IDENTIFICATION EVIDENCE

Many criminal cases revolve about the issue of identity: is the suspect S the same person as the perpetrator C of the crime? Similar issues arise in civil cases, such as disputed paternity.

Forensic trace evidence is often brought in such cases. From the crime scene we obtain information I_C that can be assumed to apply to the criminal C — thus we may have a fingerprint, a footprint, fibres, or eyewitness evidence of sex, age, race, *etc.* With advances in DNA technology, it is now common to obtain a DNA profile of the criminal from biological material left at the scene of the crime. In addition, we have similar information I_S about the suspect S, for example his DNA profile. When this matches the crime sample, *i.e.* $I_S = I_C$, $= x$, say, that is clearly evidence in favour of the two samples having the same source. But how are we properly to weigh and apply this evidence?

One relevant feature of match evidence is the *match probability* P: this is the frequency with which the characteristic x occurs in the population at large. In the case of DNA profiling, the match probability can be estimated from population figures and genetic theory. Very tiny match probabilities, even as small as one in one billion, are now routine.

3.1 THE PROSECUTOR'S FALLACY

We henceforth implicitly condition on the suspect's characteristic: $I_S = x$. The match probability can be written as $P = \Pr(I_C = x \mid C \neq S)$. If we describe this as "the probability that the crime sample came from some one other than S", we are immediately in danger of committing the prosecutor's fallacy of § 2.1, which interprets P as $\Pr(C \neq S \mid I_C = x)$, *i.e.* the probability, in the light of the match, that S is innocent — implying that the probability of guilt G is $1 - P$. If say $P = 0.0000001$, the jury or judge might well understand that the probability is only 1 in 10 million that S is not guilty, and convict.

3.2 THE DEFENCE ARGUMENT

A counter-argument along the lines of § 2.2 does not succeed here, since the probability of a match under the alternative hypothesis of guilt is unity.

Instead the defence might point out that there are $N + 1$ (say) people who could have committed this crime. One of these is truly guilty, and so matches the crime trace; while we would expect to see approximately NP innocent matches out of the remaining N innocent individual. We thus expect a total of $1 + NP$ matching individuals, of whom just 1 is guilty. If all we know about S is that he matches, the probability he is guilty is $1/(1 + NP)$. Taking $N = 30$ million and again $P = 0.0000001$, we would expect 3 innocent matches, for a final guilt probability of 1 in 4 — which is certainly not evidence "beyond a reasonable doubt".

3.3 SOME OTHER ARGUMENTS

The above defence argument can be varied in a number of ways (Dawid 1994), many of which are intuitively appealing — and have been recommended for use — but are in fact fallacious.

In all cases we assume that, prior to any evidence, any of the $N + 1$ individuals in the population is equally likely to be guilty, and that the only evidence $\mathcal{E}$ against S is that of the match: $I_S = I_C = x$. For illustration we take $N = 100$, $P = 0.004$.

Let M denote the unknown number of individuals i having $I_i = x$. We suppose that, before any samples are measured, M has the binomial distribution $\mathrm{Bin}(N+1; P)$. We have $\Pr(G \mid \mathcal{E}, M) = M^{-1}$, and the final guilt probability, $\Pr(G \mid \mathcal{E})$, can be obtained by taking the expectation of this quantity with respect to the conditional distribution of M, given the evidence $\mathcal{E}$.

1. The evidence tells us that $M \geq 1$, and simple conditioning on this yields

   $$\Pr(G \mid \mathcal{E}) = \mathrm{E}(M^{-1} \mid M \geq 1).$$

   For $M \sim \mathrm{Bin}(N+1; P)$ this is not easily expressed in closed form, but can be calculated: for our numbers it evaluates to **0.902**.

2. An alternative argument is that, given the evidence, we know that there is one guilty match, and, out of the remaining N innocent individuals,

each has, independently, probability P of supplying a match. So the conditional distribution of M is $1 + \mathrm{Bin}(N; P)$. Using this to take the expectation of M^{-1} yields

$$\Pr(G \mid \mathcal{E}) = \frac{1 - (1-P)^{N+1}}{(N+1)\,P}$$

which, for our values, gives **0.824**.

3. Finally, the correct approach.

   We can consider the total evidence $(I_C = x, I_S = x)$ as the results, both successes, of two draws, *with replacement* (since C and S could be the same individual), from the population. The probability of this, given $M = m$, is $\{m/(N+1)\}^2$ and, using Bayes's Theorem, the resulting conditional distribution of M is

$$\begin{aligned}\Pr(M = m \mid I_C = x, I_S = x) \\ = \quad c\,m \binom{N}{m-1} P^{m-1} (1-P)^{N-m+1} \\ (m = 1, \ldots, N+1),\end{aligned}$$

   where the normalising constant is $c = 1/(1+NP)$. Taking the expectation of M^{-1} with respect to this distribution then yields

$$\Pr(G \mid \mathcal{E}) = 1/(1+NP),$$

   in agreement with the original (and much simpler) defence argument. This evaluates numerically to **0.714**.

The above is just one example of the pitfalls besetting logical and probabilistic reasoning in cases at law: see Balding and Donnelly (1995); Dawid and Mortera (1995); Dawid and Mortera (1996); Dawid and Mortera (1998) for a number of other subtle issues of interpretation of forensic identification evidence.

3.4 BAYES

A serious problem with both the prosecution and the defence arguments is that they do not allow for the incorporation of any other evidence in the case. The coherent approach to combining identification and other evidence is through Bayes's Theorem: *Posterior Odds* (on G) = *Prior Odds* $\times$ *Likelihood Ratio*, where the other evidence is accounted for in the prior odds, and the likelihood ratio based on evidence $\mathcal{E}$ (where here $\mathcal{E}$ is the match evidence "$I_C = I_S = x$") is defined by:

$$LR = \frac{\Pr(\mathcal{E} \mid G)}{\Pr(\mathcal{E} \mid \overline{G})}. \qquad (1)$$

Because there is typically a subjective element in assessing prior probabilities, it is often argued that experts should confine their evidence to assessment of the more "objective" likelihood ratio, leaving the court to apply Bayes's Theorem with its own prior inputs. (However, see §§ 4 and 5 below concerning ambiguities in the definition of the likelihood ratio.)

In the case of identification evidence we can (usually) take $\Pr(\mathcal{E} \mid G) = 1$, $\Pr(\mathcal{E} \mid \overline{G}) = P$, so that the likelihood ratio is $1/P$. If the prior probability of guilt is π, the posterior probability is $\pi/(\pi + P - \pi P)$. This agrees (approximately) with the argument of the prosecutor when $\pi = 0.5$, and (exactly) with that of the defence when all $N+1$ potential culprits are *a priori* equally likely to be the guilty party. This might be seen as support for the defence argument in the absence of any other evidence.

An interesting application of Bayes's Theorem was in the 1995 trial of Denis John Adams for sexual assault. The only prosecution evidence was a DNA match, with match probability assessed between 1 in 2 million and 1 in 200 million. The defence relied on the fact that the victim did not identify Adams at an identification parade, and also said that he did not look like the man who had raped her. In addition Adams's girlfriend testified that he had been with her at the time of the crime.

On the basis that the criminal was likely to be a local male aged between about 18–60, the prior probability of guilt, before any evidence, might be assessed at around one in 200,000. The likelihood ratio based on the DNA match is $1/P = 2$ million, say. That based on the victim's non-recognition of Adams could be assessed at, say, $0.1/0.9 = 1/9$, and that based on his girlfriend's alibi at, say, $0.25/0.5 = 1/2$. Assuming suitable independence, the posterior odds on guilt become $(1/200{,}000) \times (2{,}000{,}000) \times (1/9) \times (1/2) = 5/9$, corresponding to a posterior probability of 35% (though rising to 98% if we take $P = 1$ in 200 million).

In the actual case this argument was allowed at trial (although it does not seem to have impressed the jury, who convicted), but ruled out on appeal, on the basis that explaining how to think about probabilistic evidence "usurps the function of the jury", which "must apply its common sense". Unfortunately that leaves the door wide open to the prosecutor's fallacy and other tempting but misleading arguments.

4 DATABASE SEARCH

In some cases where a DNA profile is found at the crime scene there may be no obvious suspect. Then a trawl may be made through a police computer DNA

database in the hope that it will throw up a match. Suppose this happens: how, if at all, does the fact of the database search affect the strength of the evidence against a suspect so identified?

For definiteness, suppose that the database $\mathcal{D}$ is of size $n = 10{,}000$, that the match probability of the crime profile is $P = 1$ in 1 million, and that exactly one profile — that of S, say — in the database is found to match.

One intuition is that the database search has eliminated 9,999 individuals who would otherwise have remained alternative suspects. Given the very large initial number of alternative suspects, this has the effect of rendering the evidence in favour of S's guilt *very marginally stronger*. The relevant likelihood ratio is still close to 1 million.

An entirely different intuition proceeds by analogy with frequentist statistical approaches to testing multiple hypotheses. This would adjust the match probability to take account of the 10,000 possible ways of obtaining a match in the database, replacing it by the value, close to 10,000 × (1 in 1 million) = 1/100, of the probability of finding a match in the database, if it does not include the criminal. And a match probability of only 1 in 100 is *vastly weaker* evidence than one of 1 in 1 million. In particular, it corresponds to a likelihood ratio in favour of guilt of 100, rather than 1 million.

Whereas the former intuition focuses directly on the hypothesis H_S that S is guilty, the latter addresses this issue indirectly by focusing on the hypothesis $H_{\mathcal{D}}$ that some one in the database is guilty. Since we know there was exactly one match, to S, and only a matching individual can be guilty, these two hypotheses appear logically equivalent. Were this so, there would be no strong reason to focus on one rather than the other — which would be problematic, in view of the enormous difference between their associated likelihood ratios. Stockmarr (1999) has argued in favour of $H_{\mathcal{D}}$, and thus of quoting a likelihood ratio of 100, on the grounds that this hypothesis is data-independent, whereas hypothesis H_S can not even be specified in advance of performing the search and identifying S. However, while such data-dependence can affect frequentist inferences, its relevance to likelihood inference is arguable.

In fact, although hypotheses $H_{\mathcal{D}}$ and H_S are indeed equivalent once we know that the database $\mathcal{D}$ contains exactly one match, to S, they were not equivalent before making that observation: we may term them *conditionally equivalent*. A way of bridging the apparent chasm between them appears on realising that the *prior probability* of $H_{\mathcal{D}}$ is about 10,000 times larger than that of H_S. And, as we move between these hypotheses, this factor between their prior odds cancels exactly with that between their associated likelihood ratios noted above. Both approaches thus produce the identical posterior probability (whether for H_S or for $H_{\mathcal{D}}$ being unimportant, since these have truly become logically equivalent subsequent to the database search and its findings). If, taking a fundamental Bayesian position, we regard our inference as entirely carried by the posterior probability, there is thus no incompatibility between the two analyses.

We see that, in the absence of a clearly specified hypothesis, the concept of "the likelihood ratio" can not be regarded as objectively meaningful in itself, but rather is just one, volatile, ingredient of the (invariant) posteror inference — requiring the equally and oppositely volatile prior probability to complete that inference.

While this fully Bayesian analysis resolves the conceptual paradox, a serious practical problem remains. If "objectivity" requires that we offer likelihood ratios, rather than posterior probabilities, in evidence, which should we give? — and how can we ensure that its meaning and use is properly appreciated? As a matter of psychology it seems to me preferable to quote the likelihood ratio relating to H_S: the court will surely find it easier to understand and assess the prior probability that S is guilty, which is what is then needed to complete the analysis, rather than (as required for a $H_{\mathcal{D}}$-focused analysis) the prior probability that the guilty party is in the database. This choice is also legally preferable, since the fact that S was identified by searching a database may be inadmissible as evidence.

For further (often heated) discussion of these issues see Balding and Donnelly (1996); Donnelly and Friedman (1999); Stockmarr (1999); Evett *et al.* (2000); Dawid (2001).

5 MULTIPLE PERPETRATORS AND STAINS

A similar problem (Meester and Sjerps 2003; Meester and Sjerps 2004) arises when we know there were two criminals, two distinct DNA stains (say one on a pillow, one on a sheet) have been found at the scene of the crime, and there is a single suspect, S, who matches one of them — say the pillow stain — with its associated match probability P. How is the strength of the evidence against S affected by the multiplicity of stains?

Once again there is a choice of hypotheses to compare, these being logically equivalent in the light of the findings, but not in advance. A first approach compares

"S left one of the two stains" with "S did not leave either stain'; a second compares "S left the pillow stain" with "S did not leave either stain"; and yet a third compares "S left the pillow stain" with "S did not leave the pillow stain". Under some assumptions, the associated likelihood ratios are, respectively, $\frac{1}{2}P$, P, and $\frac{1}{2}P \times (2-\delta)/(1-\delta)$, where δ is the prior probability that S is guilty. And once again, the differences between these disappear after they are combined with their varying relevant prior odds. In Dawid (2004) I argue that it is the first of these likelihood ratios that relates most directly to the relevant issue: that of the guilt of S. But one must also take into account that the knowledge that there were two culprits effectively doubles the prior probability of S's guilt, as compared with a single-suspect case.

6 MIXED STAINS

In many cases, *e.g.* involving a rape or scuffle, a crime trace may clearly[1] be a mixture of biological material from more than one individual. We may or may not know how many contributors are involved, or the identity of some of them. It is sometimes possible to separate out the components of different contributors, *e.g.* by taking into account the differing amounts of DNA at different bands, but this is unreliable.

Suppose we have a suspect S who "matches" the crime trace, in that all his bands are contained in it. What is the strength of the DNA evidence against him? This can involve complex and subtle calculations and be sensitive to assumptions made.

6.1 O. J. SIMPSON

In the celebrated trial of O. J. Simpson for double murder, one of the crime samples could be explained as a mixture of blood from Simpson and one of the victims, Ron Goldman. At a certain locus, Simpson had genotype AB, Goldman AC, and the crime sample had ABC. In pre-trial depositions[2], the prosecution argued that the relevant match probability P should be taken as the frequency of Simpson's genotype AB — about 5%. (Such a P would be multiplied by similar figures calculated for other loci to obtain an overall match probability). The defence argued that P should be the total probability of any of the genotypes, AA, AB, AC, BB, BC, CC, that would have "matched" the crime sample: about 39%.

However, on the assumption that the mixture consists of Goldman and the culprit, the culprit must have type AB, BB or BC. These have combined probability 21%, and it is the reciprocal of this figure for P that yields the correct likelihood ratio. If we did not know Goldman's genotype, or thought that the other contributor was some one else, we need to conduct a more complex calculation to obtain the relevant likelihood ratio. Interpreting this as P^{-1}, we again obtain $P \approx 21\%$ (though this is an accidental concurrence of two potentially different figures).

[1] For example, because it has more than two bands at some locus.

[2] `http://tinyurl.com/2fhsx`

7 MISSING SUSPECT

When a suspect, or other relevant party, is not available for DNA profiling, useful information can sometimes be obtained by profiling relatives — although the analysis then required can be both conceptually and computationally challenging.

7.1 HANRATTY

In 1962 James Hanratty was executed for rape and murder. In 1998 a DNA profile, assumed to be from the culprit, was extracted from some items that had been stored since the crime. Its associated match probability was around 1 in 2.5 million. Ever ready to fall for the prosecutor's fallacy, the Press duly reported this as "There is a 1 in 2.5 million chance that Hanratty was not the A6 killer" — even though, since Hanratty's DNA was unavailable, there was no more evidence against him than against any one else.

Hanratty's mother and brother now offered their own DNA for profiling — and this failed to exclude him. Again reports of the above match probability circulated as evidence of his guilt. In fact, the actual likelihood ratio, based on the indirect evidence of his relatives' DNA, was around 440.

Finally his body was exhumed, and a direct match obtained. Although the defence attempted to attribute this to contamination, it is generally agreed that the case is now closed.

7.2 DISPUTED PATERNITY

Problems of disputed paternity necessarily revolve around indirect "matching" of the DNA of the putative father with that of the true father, as partially revealed through the child's DNA. When profiles from mother, child and putative father are available, the likelihood ratio in favour of paternity can be calculated by standard formulae. When the putative father's profile is unavailable, profiles may be obtained from his relatives: for example, two full brothers, and an undisputed child and its (different) mother. Although the logical steps in calculating the likelihood

ratio are clear in principle (though not always so to the forensic and other experts directly involved in such work), the computational difficulties of implementing them can be severe.

8 BAYESIAN NETWORKS

In many court cases there is a mixed mass of evidence, the various items relating to each other and to the ultimate issue in complex and subtle ways. Whereas most lawyers are content to handle these complexities in purely intuitive ways, some legal scholars have considered formal tools to help in this process. An early and still influential contribution was the development of the *Wigmore chart* (Wigmore 1913; Wigmore 1937), a graphical representation of qualitative relationships between items of evidence.

There is some *prima facie* similarity between Wigmore charts and the modern technology of *Bayesian networks* (Cowell *et al.* 1999). Bayesian networks have been used for both qualitative and quantitative analysis of legal evidence. The former was undertaken by Dawid and Evett (1997) in the context of a prosecution for robbery, where it was required to combine eyewitness, fibre and blood evidence: the structure of the network implies various conditional independence relationships between the variables, that can be extracted and used to simplify expressions for likelihood ratios. Another fascinating use of Bayesian networks (which was also strongly influenced by Wigmorean analysis) is the reanalysis by Kadane and Schum (1996) of the evidence in the celebrated murder trial of Sacco and Vanzetti.

Bayesian networks have proved particularly valuable in addressing complex problems of interpretation of DNA profile evidence. Thus Figure 1 shows a Bayesian network representation of the absent father paternity case described in § 7.2. This displays evidential relationships in a semantically unambiguous, clear and striking manner, and supports complex computational analysis using general purpose Probabilistic Expert System software such as HUGIN[3] — see Dawid *et al.* (2002) for further details. This technology is also being applied to still more complex problems of DNA analysis, involving, separately or in combination, such features as mixed stains (Mortera *et al.* 2003), mutation (Dawid *et al.* 2001; Dawid 2003), contamination, and field and laboratory errors.

[3] `http://www.hugin.com`

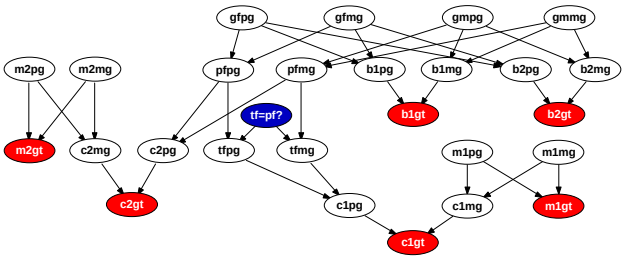


Figure 1: Bayesian Network Representation of a Complex Paternity Case

9 CONCLUSIONS

Seemingly straightforward problems of legal reasoning can quickly lead to complexity, controversy and confusion: the above examples are just a few amongst many. The field provides a rich and challenging testbed for any general approach or technique for reasoning under uncertainty. Success in this venture could also have real impact on the fairer administration of justice.

Acknowledgements

The work reported here was funded in part by the Gatsby Charitable Foundation and the Leverhulme Trust.

References

Balding, D. J. and Donnelly, P. J. (1995). Inference in forensic identification (with Discussion). *Journal of the Royal Statistical Society, Series A*, **158**, 21–53.

Balding, D. J. and Donnelly, P. J. (1996). DNA profile evidence when the suspect is identified through a database search. *Journal of Forensic Sciences*, **41**, 603–7.

Brilmayer, L. and Kornhauser, L. (1978). Quantitative methods and legal decisions. *University of Chicago Law Review*, **46**, 116–53.

Cohen, L. J. (1977). *The Probable and the Provable.* Oxford : Clarendon Press.

Cohen, L. J. (1980). The logic of proof. *Criminal Law Review*, **1980**, 91–103.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. (1999). *Probabilistic Networks and Expert Systems.* Springer, New York.

Dawid, A. P. (1994). The island problem: Coherent use of identification evidence. In *Aspects of Uncertainty: A Tribute to D. V. Lindley*, (ed. P. R. Freeman and A. F. M. Smith), chapter 11, pp. 159–70. John Wiley and Sons, Chichester.

Dawid, A. P. (2001). Comment on Stockmarr's "Likelihood ratios for evaluating DNA evidence, when the suspect is found through a database search" (with response by Stockmarr). *Biometrics*, **57**, 976–80.

Dawid, A. P. (2002). Bayes's theorem and weighing evidence by juries. *Proceedings of the British Academy*, **113**, 71–90. (Published in book form as *Bayes's Theorem*, edited by Richard Swinburne).

Dawid, A. P. (2003). An object-oriented Bayesian network for estimating mutation rates. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, Jan 3–6 2003, Key West, Florida,* (ed. C. M. Bishop and B. J. Frey). `http://tinyurl.com/39bmh`.

Dawid, A. P. (2004). Which likelihood ratio? In discussion of Meester and Sjerps (2004). *Law, Probability and Risk*, **3**, 65–71.

Dawid, A. P. and Evett, I. W. (1997). Using a graphical method to assist the evaluation of complicated patterns of evidence. *Journal of Forensic Sciences*, **42**, 226–31.

Dawid, A. P. and Mortera, J. (1995). In discussion of Balding and Donnelly (1995). *Journal of the Royal Statistical Society, Series A*, **158**, 46.

Dawid, A. P. and Mortera, J. (1996). Coherent analysis of forensic identification evidence. *Journal of the Royal Statistical Society, Series B*, **58**, 425–43.

Dawid, A. P. and Mortera, J. (1998). Forensic identification with imperfect evidence. *Biometrika*, **85**, 835–49.

Dawid, A. P., Mortera, J., and Pascali, V. L. (2001). Non-fatherhood or mutation? A probabilistic approach to parental exclusion in paternity testing. *Forensic Science International*, **124**, 55–61.

Dawid, A. P., Mortera, J., Pascali, V. L., and van Boxel, D. W. (2002). Probabilistic expert systems for forensic inference from genetic markers. *Scandinavian Journal of Statistics*, **29**, 577–95.

Donnelly, P. J. and Friedman, R. D. (1999). DNA database searches and the legal consumption of scientific evidence. *Michigan Law Review*, **97**, 931–84.

Eggleston, R. (1980). The probability debate. *Criminal Law Review*, **1980**, 678–88.

Evett, I. W., Foreman, L. A., and Weir, B. S. (2000). Letter to the Editor (with responses by A. Stockmarr and B. Devlin). *Biometrics*, **56**, 1274–5.

Fairley, W. B. and Mosteller, F. (1977). A conversation about Collins. In *Statistics in Public Policy*, (ed. W. B. Fairley and F. Mosteller), pp. 369–79. Addison-Wesley, Reading, Massachusetts.

Finkelstein, M. O. and Fairley, W. B. (1970). A Bayesian approach to identification evidence. *Harvard Law Review*, **83**, 489–517.

Finkelstein, M. O. and Fairley, W. B. (1971). A comment on "Trial by Mathematics". *Harvard Law Review*, **84**, 1801–9.

Gigerenzer, G. (2002). *Reckoning with Risk: Learning to Live with Uncertainty.* Allen Lane: The Penguin Press.

Kadane, J. B. and Schum, D. A. (1996). *A Probabilistic Analysis of the Sacco and Vanzetti Evidence.* John Wiley and Sons, New York.

Kaye, D. (1979). The laws of probability and the law of the land. *University of Chicago Law Review*, **47**, 34–56.

Meester, R. W. J. and Sjerps, M. (2003). The evidential value in the DNA database search controversy and the two stain problem. *Biometrics*, **59**, 727–32.

Meester, R. W. J. and Sjerps, M. (2004). Why the effect of prior odds should accompany the likelihood ratio when reporting DNA evidence (with discusion by A. P. Dawid, D. J. Balding, J. S. Buckleton and C. M. Triggs). *Law, Probability and Risk*, **3**, 51–86.

Mortera, J., Dawid, A. P., and Lauritzen, S. L. (2003). Probabilistic expert systems for DNA mixture profiling. *Theoretical Population Biology*, **63**, 191–205.

Schum, D. (1979). A review of the case against Blaise Pascal and his heirs. *Michigan Law Review*, **77**, 446–83.

Stockmarr, A. (1999). Likelihood ratios for evaluating DNA evidence when the suspect is found through a database search. *Biometrics*, **55**, 671–7.

Tribe, L. H. (1971a). A further critique of mathematical proof. *Harvard Law Review*, **84**, 1810–20.

Tribe, L. H. (1971b). Trial by mathematics: Precision and ritual in the legal process. *Harvard Law Review*, **84**, 1329–93.

Wigmore, J. H. (1913). The problem of proof. *Illinois Law Review*, **8**, 77–99.

Wigmore, J. H. (1937). *The Science of Judicial Proof*, (third edn). Little, Brown, Boston.

Williams, G. (1979). The mathematics of proof. *Criminal Law Review*, **1979**, 297–308 and 340–54.

Williams, G. (1980). A short rejoinder. *Criminal Law Review*, **1980**, 103–7.

Efficient Non-Parametric Function Induction in Semi-Supervised Learning

Olivier Delalleau, Yoshua Bengio and Nicolas Le Roux
Dept. IRO, Université de Montréal
P.O. Box 6128, Succ. Centre-Ville, Montreal, H3C 3J7, Qc, Canada
{delallea,bengioy,lerouxni}@iro.umontreal.ca

Abstract

There has been an increase of interest for semi-supervised learning recently, because of the many datasets with large amounts of unlabeled examples and only a few labeled ones. This paper follows up on proposed non-parametric algorithms which provide an estimated continuous label for the given unlabeled examples. First, it extends them to function induction algorithms that minimize a regularization criterion applied to an out-of-sample example, and happen to have the form of Parzen windows regressors. This allows to predict test labels without solving again a linear system of dimension n (the number of unlabeled and labeled training examples), which can cost $O(n^3)$. Second, this function induction procedure gives rise to an efficient approximation of the training process, reducing the linear system to be solved to $m \ll n$ unknowns, using only a subset of m examples. An improvement of $O(n^2/m^2)$ in time can thus be obtained. Comparative experiments are presented, showing the good performance of the induction formula and approximation algorithm.

1 INTRODUCTION

Several non-parametric approaches to semi-supervised learning (see (Seeger, 2001) for a review of semi-supervised learning) have been recently introduced, e.g. in (Szummer & Jaakkola, 2002; Chapelle et al., 2003; Belkin & Niyogi, 2003; Zhu et al., 2003a; Zhu et al., 2003b; Zhou et al., 2004). They rely on weak implicit assumptions on the generating data distribution, e.g. smoothness of the target function with respect to a given notion of similarity between examples[1]. For classification tasks this amounts to assuming that the target function is constant within the region of input space (or "cluster" (Chapelle et al., 2003)) associated with a particular class. These previous non-parametric approaches exploit the idea of building and smoothing a graph in which each example is associated with a node, and arcs between two nodes are associated with the value of a similarity function applied on the corresponding two examples.

It is not always clear with these graph-based kernel methods for semi-supervised learning how to generalize to previously unseen test examples. In general they have been designed for the transductive setting, in which the test examples must be provided before doing the expensive part of training. This typically requires solving a linear system with n equations and n parameters, where n is the number of labeled and unlabeled data. In a truly inductive setting where new examples are given one after the other and a prediction must be given after each example, it can be very computationally costly to solve such a system anew for each of these test examples. In (Zhu et al., 2003b) it is proposed to assign to the test case the label (or inferred label) of the nearest neighbor (NN) from the training set (labeled or unlabeled). In this paper we derive from the training criterion an inductive formula that turns out to have the form of a Parzen windows predictor, for a computational cost that is $O(n)$. Besides being smoother than the NN-algorithm, this induction formula is consistent with the predicted labels on the unlabeled training data.

In addition to providing a relatively cheap way of doing function induction, the proposed approach opens the door to efficient approximations even in the transductive setting. Since we know the analytic functional form of the prediction at a point x in terms of the predictions at a set of training points, we can use it to express all the predictions in terms of a small subset of $m \ll n$ examples (i.e. a low-rank approximation) and solve a linear system with m variables and equations.

[1] See also (Kemp et al., 2004) for a hierarchically structured notion of a priori similarity.

2 NON-PARAMETRIC SMOOTHNESS CRITERION

In the mathematical formulations, we only consider here the case of binary classification. Each labeled example x_k ($1 \quad k \quad l$) is associated with a label $y_k \in \{-1, 1\}$, and we turn the classification task into a regression one by looking for the values of a function f on both labeled and unlabeled examples x_i ($1 \quad i \quad n$), such that $f(x_i) \in [-1, 1]$. The predicted class of x_i is thus $sign(f(x_i))$. Note however that all algorithms proposed extend naturally to multiclass problems, using the usual one vs. rest trick.

Among the previously proposed approaches, several can be cast as the minimization of a criterion (often a quadratic form) in terms of the function values $f(x_i)$ at the labeled and unlabeled training examples x_i:

$$C_{W,D,D',\lambda}(f) = \frac{1}{2} \sum_{i,j \in U \cup L} W(x_i, x_j) D(f(x_i), f(x_j)) + \lambda \sum_{i \in L} D'(f(x_i), y_i) \quad (1)$$

where U is the unlabeled set, L the labeled set, x_i the i-th example, y_i the target label for $i \in L$, $W(\cdot, \cdot)$ is a positive similarity function (e.g. a Gaussian kernel) applied on a pair of inputs, and $D(\cdot, \cdot)$ and $D'(\cdot, \cdot)$ are lower-bounded dissimilarity functions applied on a pair of output values. Three methods using a criterion of this form have already been proposed: (Zhu et al., 2003a), (Zhou et al., 2004) (where an additional regularization term is added to the cost, equal to $\lambda \sum_{i \in U} f(x_i)^2$), and (Belkin et al., 2004) (where for the purpose of theoretical analysis, they add the constraint $\sum_i f(x_i) = 0$). To obtain a quadratic form in $f(x_i)$ one typically chooses D and D' to be quadratic, e.g. the Euclidean distance. This criterion can then be minimized exactly for the n function values $f(x_i)$. In general this could cost $O(n^3)$ operations, possibly less if the input similarity function $W(\cdot, \cdot)$ is sparse.

A quadratic dissimilarity function makes a lot of sense in regression problems but has also been used successfully in classification problems, by looking for a *continuous* labeling function f. The first term of eq. 1 indeed enforces the smoothness of f. The second term makes f consistent with the given labels. The hyperparameter λ controls the trade-off between those two costs. It should depend on the amount of noise in the observed values y_i, i.e. on the particular data distribution (although for example (Zhu et al., 2003a) consider forcing $f(x_i) = y_i$, which corresponds to $\lambda = +\infty$).

In the following we study the case where D and D' are the Euclidean distance. We also assume samples are sorted so that $L = \{1, \ldots, l\}$ and $U = \{l+1, \ldots, n\}$. The minimization of the criterion w.r.t. all the $f(x_i)$ for $i \in L \cup U$ then gives rise to the linear system

$$A\vec{f} = \lambda \vec{y} \quad (2)$$

with

$$\vec{y} = (y_1, \ldots, y_l, 0, \ldots, 0)^T \quad (3)$$

$$\vec{f} = (f(x_1), \ldots, f(x_n))^T$$

and, using the matrix notation $W_{ij} = W(x_i, x_j)$, the matrix A written as follows:

$$A = \lambda \Delta_L + Diag(W\mathbf{1}_n) - W \quad (4)$$

where $Diag(v)$ is the matrix whose diagonal is the vector v, $\mathbf{1}_n$ is the vector of n ones, and Δ_L ($n \times n$) is

$$(\Delta_L)_{ij} = \delta_{ij} \delta_{i \in L}. \quad (5)$$

This solution has the disadvantage of providing no obvious prediction for new examples, but the method is generally used transductively (the test examples are included in the unlabeled set). To obtain function induction without having to solve the linear system for each new test point, one alternative would be to parameterize f with a flexible form such as a neural network or a linear combination of non-linear bases (see also (Belkin & Niyogi, 2003)). Another is the induction formula proposed below.

3 FUNCTION INDUCTION FORMULA

In order to transform the above transductive algorithms into function induction algorithms we will do two things: (i) consider the same type of smoothness criterion as in eq. 1, but including a test example x, and (ii) as in ordinary function induction (by opposition to transduction), require that the value of $f(x_i)$ on training examples x_i remain fixed even after x has been added[2].

The second point is motivated by the prohibitive cost of solving again the linear system, and the reasonable assumption that the value of the function over the unlabeled examples will not change much with the addition of a new point. This is clearly true asymptotically (when $n \to \infty$). In the non-asymptotic case we should expect transduction to perform better than induction (again, assuming test samples are drawn from the same distribution as the training data), but as shown in our experiments, the loss is typically very small, and comparable to the variability due to the selection of training examples.

Adding terms for a new unlabeled point x in eq. 1 and keeping the value of f fixed on the training points x_j

[2]Here we assume x to be drawn from the same distribution as the training samples: if it is not the case, this provides another justification for keeping the $f(x_i)$ fixed.

leads to the minimization of the modified criterion

$$C^*_{W,D}(f(x)) = \sum_{j \in U \cup L} W(x, x_j) D(f(x), f(x_j)). \quad (6)$$

Taking for D the usual Euclidean distance, $C^*_{W,D}$ is convex in $f(x)$ and is minimized when

$$f(x) = \frac{\sum_{j \in U \cup L} W(x, x_j) f(x_j)}{\sum_{j \in U \cup L} W(x, x_j)} = \tilde{f}(x). \quad (7)$$

Interestingly, this is exactly the formula for Parzen windows or Nadaraya-Watson non-parametric regression (Nadaraya, 1964; Watson, 1964) when W is the Gaussian kernel and the estimated $f(x_i)$ on the training set are considered as desired values.

One may want to see what happens when we apply $\tilde{f}$ on a point x_i of the training set. For $i \in U$, we obtain that $\tilde{f}(x_i) = f(x_i)$. But for $i \in L$,

$$\tilde{f}(x_i) \quad = \quad f(x_i) + \frac{\lambda(f(x_i) - y_i)}{\sum_{j \in U \cup L} W(x_i, x_j)}.$$

Thus the induction formula (eq. 7) gives the same result as the transduction formula (implicitely defined by eq. 2) over unlabeled points, but on labeled examples it chooses a value that is "smoother" than $f(x_i)$ (not as close to y_i). This may lead to classification errors on the labeled set, but generalization error may improve by allowing non-zero training error on labeled samples. This remark is also valid in the special case where $\lambda = +\infty$, where we fix $f(x_i) = y_i$ for $i \in L$, because the value of $\tilde{f}(x_i)$ given by eq. 7 may be different from y_i (though experiments showed such label changes were very unlikely in practice).

The proposed algorithm for semi-supervised learning is summarized in algorithm 1, where we use eq. 2 for training and eq. 7 for testing.

Algorithm 1 Semi-supervised induction

(1) Training phase

Compute $A = \lambda \Delta_L + Diag(W\mathbf{1}_n) - W$ (eq. 4)

Solve the linear system $A\vec{f} = \lambda \vec{y}$ (eq. 2) to obtain $f(x_i) = \vec{f_i}$

(2) Testing phase

For a new point x, compute its label $\tilde{f}(x)$ by eq. 7

4 SPEEDING UP THE TRAINING PHASE

A simple way to reduce the cubic computational requirement and quadratic memory requirement for 'training' the non-parametric semi-supervised algorithms of section 2 is to force the solutions to be expressed in terms of a **subset of the examples**. This idea has already been exploited successfully in a different form for other kernel algorithms, e.g. for Gaussian processes (Williams & Seeger, 2001).

Here we will take advantage of the induction formula (eq. 7) to simplify the linear system to $m \ll n$ equations and variables, where m is the size of a subset of examples that will form a basis for expressing all the other function values. Let $S \subset L \cup U$ with $L \subset S$ be such a subset, with $|S| = m$. Define $R = U \backslash S$. The idea is to force $f(x_i)$ for $i \in R$ to be expressed as a linear combination of the $f(x_j)$ with $j \in S$:

$$\forall i \in R, f(x_i) = \frac{\sum_{j \in S} W(x_i, x_j) f(x_j)}{\sum_{j \in S} W(x_i, x_j)}. \quad (8)$$

Plugging this in eq. 1, we separate the cost in four terms $(C_{RR}, C_{RS}, C_{SS}, C_L)$:

$$\begin{aligned} & \underbrace{\frac{1}{2} \sum_{i,j \in R} W(x_i, x_j) \left(f(x_i) - f(x_j)\right)^2}_{C_{RR}} \\ + & \underbrace{2 \times \frac{1}{2} \sum_{i \in R, j \in S} W(x_i, x_j) \left(f(x_i) - f(x_j)\right)^2}_{C_{RS}} \\ + & \underbrace{\frac{1}{2} \sum_{i,j \in S} W(x_i, x_j) \left(f(x_i) - f(x_j)\right)^2}_{C_{SS}} \\ + & \underbrace{\lambda \sum_{i \in L} (f(x_i) - y_i)^2}_{C_L} \end{aligned}$$

Let $\vec{f}$ denote now the vector with entries $f(x_i)$, only for $i \in S$ (they are the values to identify). To simplify the notations, decompose W in the following sub-matrices:

$$W = \begin{array}{cc} W_{SS} & W'_{RS} \\ W_{RS} & W_{RR} \end{array}.$$

with W_{SS} of size $(m \times m)$, W_{RS} of size $((n-m) \times m)$ and W_{RR} of size $((n-m) \times (n-m))$. Also define $\overline{W}_{RS}$ the matrix of size $((n-m) \times m)$ with entries $\frac{W_{ij}}{\sum_{k \in S} W_{ik}}$, for $i \in R$ and $j \in S$.

Using these notations, the gradient of the above cost with respect to $\vec{f}$ can be written as follows:

$$\begin{aligned} & \underbrace{\left[2\left(\overline{W}^T_{RS}\left(Diag(W_{RR}\mathbf{1}_r) - W_{RR}\right)\overline{W}_{RS}\right)\right]\vec{f}}_{\frac{\partial C_{RR}}{\partial \vec{f}}} \\ + & \underbrace{\left[2\left(Diag(W_{SR}\mathbf{1}_r) - \overline{W}^T_{RS} W_{RS}\right)\right]\vec{f}}_{\frac{\partial C_{RS}}{\partial \vec{f}}} \\ + & \underbrace{\left[2\left(Diag(W_{SS}\mathbf{1}_m) - W_{SS}\right)\right]\vec{f}}_{\frac{\partial C_{SS}}{\partial \vec{f}}} + \underbrace{2\lambda\Delta_L(\vec{f} - \vec{y})}_{\frac{\partial C_L}{\partial \vec{f}}} \end{aligned}$$

where Δ_L is the same as in eq. 5, but is of size $(m \times m)$, and $\vec{y}$ is the vector of targets (eq. 3), of size m. The

linear system $A\vec{f} = \lambda\vec{y}$ of eq. 2 is thus redefined with the following system matrix:

$$\begin{aligned} A = \quad & \lambda\Delta_L \\ + \quad & \overline{W}_{RS}^T \left(Diag(W_{RR}\mathbf{1}_r) - W_{RR}\right)\overline{W}_{RS} \\ + \quad & Diag(W_{SR}\mathbf{1}_r) - \overline{W}_{RS}^T W_{RS} \\ + \quad & Diag(W_{SS}\mathbf{1}_m) - W_{SS}. \end{aligned}$$

The main computational cost now comes from the computation of $\frac{\partial C_{RR}}{\partial \vec{f}}$. To avoid it, we simply choose to ignore C_{RR} in the total cost, so that the matrix A can be computed in $O(m^2(n-m))$ time, using only $O(m^2)$ memory, instead of respectively $O(m(n-m)^2)$ time and $O(m(n-m))$ memory when keeping C_{RR}. By doing so we lessen the smoothness constraint on f, since we do not take into account the part of the cost enforcing smoothness between the examples in R. However, this may have a beneficial effect. Indeed, the costs C_{RS} and C_{RR} can be seen as regularizers encouraging the smoothness of f on R. In particular, using C_{RR} may induce strong constraints on f that could be inappropriate when the approximation of eq. 8 is inexact (which especially happens when a point in R is far from all examples in S). This could constrain f too much, thus penalizing the classification performance. In this case, discarding C_{RR}, besides yielding a significant speed-up, also gives better results. Algorithm 2 summarizes this algorithm (not using C_{RR}).

Algorithm 2 Fast semi-supervised induction

Choose a subset $S \supseteq L$ (e.g. with algorithm 3)
$R \leftarrow U \setminus S$
(1) Training phase

$$\begin{aligned} A \leftarrow \quad & \lambda\Delta_L + Diag(W_{SR}\mathbf{1}_r) \\ - \quad & \overline{W}_{RS}^T W_{RS} + Diag(W_{SS}\mathbf{1}_m) - W_{SS} \end{aligned}$$

Solve the linear system $A\vec{f} = \lambda\vec{y}$ to obtain $f(x_i) = \vec{f}_i$ for $i \in S$
Use eq. 8 to obtain $f(x_i)$ for $i \in R$
(2) Testing phase
For a new point x, compute its label $\tilde{f}(x)$ by eq. 7

In general, training using only a subset of $m \ll n$ samples will not perform as well as using the whole dataset. Thus, it can be important to choose the examples in the subset carefully to get better results than a random selection. Our criterion to choose those examples is based on eq. 8, that shows $f(x_i)$ for $i \notin S$ should be well approximated by the value of f at the neighbors of x_i in S (the notion of neighborhood being defined by W). Thus, in particular, x_i for $i \notin S$ should not be too far from the examples in S. This is also important because when discarding the part C_{RR} of the cost, we must be careful to cover the whole manifold with S, or we may leave "gaps" where the smoothness of f would not be enforced. This suggests to start with $S = \emptyset$ and $R = U$, then add samples x_i iteratively by choosing the point farthest from the current subset, i.e. the one that minimizes $\sum_{j \in L \cup S} W(x_i, x_j)$. Note that adding a sample that is far from all other examples in the dataset will not help, thus we discard an added point if this is the case (x_j being "far" is defined by a threshold on $\sum_{i \in R \setminus \{j\}} W(x_i, x_j)$). In the worst case, this could make the algorithm in $O(n^2)$, but assuming only few examples are far from all others, it scales as $O(mn)$. Once this first subset is selected, we refine it by training the algorithm presented in section 2 on the subset S, in order to get an approximation of the $f(x_i)$ for $i \in S$, and by using the induction formula of section 3 (eq. 7) to get an approximation of the $\tilde{f}(x_j)$ for $j \in R$. We then discard samples in S for which the confidence in their labels is high[3], and replace them with samples in R for which the confidence is low (samples near the decision surface). One should be careful when removing samples, though: we make sure we do not leave "empty" regions (i.e. $\sum_{i \in L \cup S} W(x_i, x_j)$ must stay above some threshold for all $j \in R$). Finally, labeled samples are added to S. Overall, the cost of this selection phase is on the order of $O(mn + m^3)$. Experiments showing its effectiveness are presented in section 5.3. The subset selection algorithm[4] is summarized in algorithm 3.

5 EXPERIMENTS

5.1 FUNCTION INDUCTION

Here, we want to validate our induction formula (eq. 7): the goal is to show that it gives results close to what would have been obtained if the test points had been included in the training set (transduction). Indeed, we expect that the more unlabeled points the better, but how much better? Experiments have been performed on the "Letter Image Recognition" dataset of the UCI Machine Learning repository (UCI MLR). There are 26 handwritten characters classes, to be discriminated using 16 geometric features. However, to keep things simple, we reduce to a binary problem by considering only the class formed by the characters 'I' and 'O' and the class formed by 'J' and 'Q' (the choice of these letters makes the problem harder than a basic two-character classification task). This yields a dataset of 3038 samples. We use for $W(x, y)$ the Gaussian kernel with bandwidth 1: $W(x, y) = e^{-||x-y||^2}$.

First, we analyze how the labels can vary between in-

[3] In a binary classification task, the confidence is given by $|f(x_i)|$. In the multi-class case, it is the difference between the weights of the two classes with highest weights.

[4] Note that it would be interesting to investigate the use of such an algorithm in cases where one can obtain labels, but at a cost, and needs to select which samples to label.

Algorithm 3 Subset selection

δ is a small threshold, e.g. $\delta = 10^{-10}$
(1) Greedy selection
$S \leftarrow \emptyset$ {The subset we are going to build}
$R \leftarrow U$ {The rest of the unlabeled data}
while $|S| + |L| < m$ **do**
  Find $j \in R$ s.t. $\sum_{i \in R \setminus \{j\}} W(x_i, x_j) \geq \delta$ and $\sum_{i \in L \cup S} W(x_i, x_j)$ is minimum
  $S \leftarrow S \cup \{j\}$
  $R \leftarrow R \setminus \{j\}$
(2) Improving the decision surface
Compute an approximate of $f(x_i)$, $i \in S$ and $\tilde{f}(x_j)$, $j \in R$, by applying algorithm 1 with the labeled set L and the unlabeled set S and using eq. 7 on R
$S_H \leftarrow$ the points in S with highest confidence (see footnote 3)
$R_L \leftarrow$ the points in R with lowest confidence
for all $j \in S_H$ **do**
  if $\min_{i \in R} \sum_{k \in L \cup S \setminus \{j\}} W(x_i, x_k) \geq \delta$ **then**
    $k^* \leftarrow \text{argmin}_{k \in R_L} \sum_{i \in L \cup S} W(x_k, x_i)$
    $S \leftarrow (S \setminus \{j\}) \cup \{k^*\}$ {Replace j by k^* in S}
    $R \leftarrow (R \setminus \{k^*\}) \cup \{j\}$ {Replace k^* by j in R}
$S \leftarrow S \cup L$ {Add the labeled data to the subset}

duction and transduction when the test set is large (section 5.1.1), then we study how this variation compares to the intrinsic variability due to the choice of training data (section 5.1.2).

5.1.1 Induction vs. Transduction

When we add new points to the training set, two questions arise. First, do the $f(x_i)$ change significantly? Second, how important is the difference between induction and transduction over a large amount of new points, in terms of classification performance?

The experiments shown in fig. 1 have been made considering three training sets, $T1000$, $T2000$ and $T3038$, containing respectively 1000, 2000 and 3038 samples (the results plotted are averaged on 10 runs with randomly selected $T1000$ and $T2000$). The first two curves show the percentage of unlabeled data in $T1000$ and $T2000$ whose label has changed compared to the labels obtained when training over $T3038$ (the whole dataset). This validates our hypothesis that the $f(x_i)$ do not change much when adding new training points.

The next three curves show the classification error for the unlabeled data respectively on $T3038 \setminus T2000$, $T3038 \setminus T1000$ and $T3038 \setminus T1000$, for the algorithm trained respectively on $T2000$, $T1000$ and $T3038$. This allows us to see that the induction's performance is close to that of transduction (the average relative increase in classification error compared to transduction is about 20% for $T1000$ and 10% for $T2000$). In addition, the difference is very small for large amounts of labeled data as well as for very small amounts. This can be explained in the first case by the fact that enough information is available in the labeled data to get close to optimal classification, and in the second case, that there are too few labeled data to ensure an efficient prediction, either transductive or inductive.

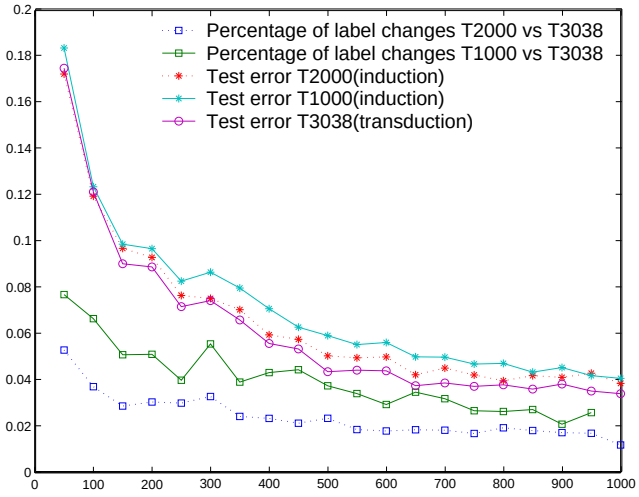


Figure 1: Percentage of unlabeled training data whose label has changed when test points were added to the training set, and classification error in induction and transduction. Horizontal axis: number of labeled data.

5.1.2 Varying The Test Set Size

The previous experiments have shown that when the test set is large in comparison with the training set, the induction formula will not be as efficient as transduction. It is thus interesting to see how evolves the difference between induction and transduction as the test set size varies in proportion with the training set size. In particular, for which size of the test set is that difference comparable to the sensitivity of the algorithm with respect to the choice of training set?

To answer this question, we need a large enough dataset to be able to choose random training sets. The whole Letters dataset is thus used here, and the binary classification problem is to discriminate the letters 'A' to 'M' from the letters 'N' to 'Z'. We take a fixed test set of size 1000. We repeat 10 times the experiments that consists in: (i) choosing a random base training set of 2000 samples (with 10% labeled), and (ii) computing the average error on test points in *transduction* by adding a fraction of them to this base training set and solving the linear system (eq. 2), repeating this so as to compute the error on all test points.

The results are shown in fig. 2, when we vary the number of test points added to the training set. Adding 0 test points is slightly different, since it corresponds to

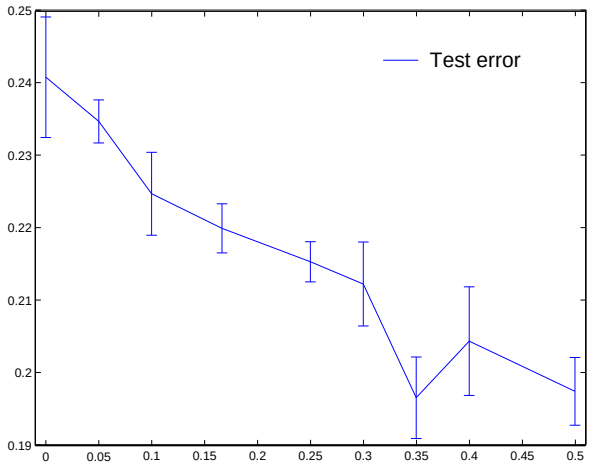


Figure 2: Horizontal axis: number of test points added in proportion with the size of the training set. Vertical axis: test error (in transduction for a proportion > 0, and in induction for the proportion 0).

the induction setting, that we plot here for comparison purpose. We see that adding a fraction of test examples corresponding to less than 5% of the training set does not yield a significant decrease in the test error compared to induction, given the intrinsic variability due to the choice of training set. It could be interesting to compare induction with the limit case where we add only 1 test point at step (ii). We did not do it because of the computational costs, but one would expect the difference with induction to be smaller than for the 5% fraction.

5.2 COMPARISON WITH EXISTING ALGORITHM

We compare our proposed algorithm (alg. 1) to the semi-supervised *Laplacian* algorithm from (Belkin & Niyogi, 2003), for which classification accuracy on the MNIST database of handwritten digits is available. Benchmarking our induction algorithm against the *Laplacian* algorithm is interesting because the latter does not fall into the general framework of section 2.

In order to obtain the best performance, a few refinements are necessary. First, it is better to use a sparse weighting function, which allows to get rid of the noise introduced by far-away examples, and also makes computations faster. The simplest way to do this is to combine the original weighting function (the Gaussian kernel) with k-nearest-neighbors. We define a new weighting function W_k by $W_k(x_i, x_j) = W(x_i, x_j)$ if x_i is a k-nearest-neighbor of x_j or vice-versa, and 0 otherwise. Second, W_k is normalized as in Spectral Clustering (Ng et al., 2002), i.e.

$$\overline{W}_k(x_i, x_j) = \frac{W_k(x_i, x_j)}{\sqrt{\frac{1}{n}\sum_{r\neq i} W_k(x_i, x_r) \sum_{r\neq j} W_k(x_r, x_k)}}.$$

Table 1: Comparative Classification Error of the *Laplacian* Algorithm (Belkin & Niyogi, 2003), $WholeSet$ in Transduction and $WholeSet$ in Induction on the MNIST Database. On the horizontal axis is the number of labeled examples and we use two different sizes of training sets (1000 and 10000 examples).

Labeled	**50**	**100**	**500**	**1000**	**5000**
Total: **1000**					
Laplacian	29.3	19.6	11.5		
$WholeSet_{trans}$	25.4	17.3	9.5		
$WholeSet_{ind}$	26.3	18.8	11.3		
Total: **10000**					
Laplacian	25.5	10.7	6.2	5.7	4.2
$WholeSet_{trans}$	25.1	11.3	5.3	5.2	3.5
$WholeSet_{ind}$	25.1	11.3	5.7	5.1	4.2

Finally, the dataset is systematically preprocessed by a Principal Component Analysis on the training part (labeled and unlabeled), to further reduce noise in the data (we keep the first 45 principal components).

Results are presented in table 1. Hyperparameters (number of nearest neighbors and kernel bandwidth) were optimized on a validation set of 20000 samples, while the experiments were done on the rest (40000 samples). The classification error is averaged over 100 runs, where the train (1000 or 10000 samples) and test sets (5000 samples) are randomly selected among those 40000 samples. Standard errors (not shown) are all under 2% of the error. The *Laplacian* algorithm was used in a transductive setting, while we separate the results for our algorithm into $WholeSet_{trans}$ (error on the training data) and $WholeSet_{ind}$ (error on the test data, obtained thanks to the induction formula)[5]. On average, both $WholeSet_{trans}$ and $WholeSet_{ind}$ slightly outperform the *Laplacian* algorithm.

5.3 APPROXIMATION ALGORITHMS

The aim of this section is to compare the classification performance of various algorithms:

> $WholeSet$, the original algorithm presented in sections 2 and 3, where we make use of all unlabeled training data (same as $WholeSet_{ind}$ in the previous section),
>
> $RSub_{subOnly}$, the algorithm that consists in speeding-up training by using only a random subset of the unlabeled training samples (the rest is completely discarded),
>
> $RSub_{RR}$ and $RSub_{noRR}$, the approximation algorithms described in section 4, when the subset is selected randomly (the second algorithm discards the part C_{RR} of the cost for faster training),

[5]See section 5.3 for the origin of the name $WholeSet$.

Table 2: Comparative Computational Requirements (n = number of training data, m = subset size)

	Time	**Memory**
$WholeSet$	$O(n^3)$	$O(n^2)$
$RSub_{subOnly}$	$O(m^3)$	$O(m^2)$
$RSub_{RR}$ $SSub_{RR}$	$O(m(n-m)^2)$	$O(m(n-m))$
$RSub_{noRR}$ $SSub_{noRR}$	$O(m^2(n-m))$	$O(m^2)$

$SSub_{RR}$ and $SSub_{noRR}$, which are similar to those above, except that the subset is now selected as in algorithm 3.

Table 2 summarizes time and memory requirements for these algorithms: in particular, the approximation method described in section 4, when we discard the part C_{RR} of the cost ($RSub_{noRR}$ and $SSub_{noRR}$), improves the computation time and memory usage by a factor approximately $(n/m)^2$.

The classification performance of these algorithms was compared on three multi-class problems: **LETTERS** is the "Letter Image Recognition" dataset from the UCI MLR. (26 classes, dimension 16), **MNIST** contains the first 20000 samples of the MNIST database of handwritten digits (10 classes, dimension 784), and **COVTYPE** contains the first 20000 samples of the normalized[6] "Forest CoverType" dataset from the UCI MLR. (7 classes, dimension 54).

We repeat 50 times the experiment that consists in choosing randomly 10000 samples as training data and the rest as the test set, and computing the test error (using the induction formula) for the different algorithms. The average classification error on the test set (with standard error) is presented in table 3 for $WholeSet$, $RSub_{subOnly}$, $RSub_{noRR}$ and $SSub_{noRR}$. For each dataset, results for a labeled fraction of 1%, 5% and 10% of the training data are presented. In algorithms using only a subset of the unlabeled data (i.e. all but $WholeSet$), the subset contains only 10% of the unlabeled set. Hyperparameters have been roughly estimated and remain fixed on each dataset. In particular, λ (in eq. 1) is set to 100 for all datasets, and the bandwidth of the Gaussian kernel used is set to 1 for LETTERS, 1.4 for MNIST and 1.5 for COVTYPE.

The approximation algorithms using the part C_{RR} of the cost ($RSub_{RR}$ and $SSub_{RR}$) are not shown in the results, because it turns out that using C_{RR} does not necessarily improve the classification accuracy, as argued in section 4. Additionally, discarding C_{RR} makes training significantly faster. Compared to $WholeSet$, typical training times with these specific settings show that $RSub_{subOnly}$ is about 150 times faster, $RSub_{noRR}$ about 15 times, and $SSub_{noRR}$ about 10 times. Note however that these factors increase very fast with the size of the dataset (10000 samples is still "small").

[6]Scaled so that each feature has standard deviation 1.

Table 3: Comparative Classification Error (Induction) of $WholeSet$, $RSub_{subOnly}$, $RSub_{noRR}$ and $SSub_{noRR}$, for Various Fractions of Labeled Data.

% labeled	LETTERS		MNIST		COVTYPE	
1%						
$WholeSet$	56.0	0.4	35.8	1.0	47.3	1.1
$RSub_{subOnly}$	59.8	0.3	29.6	0.4	**44.8**	0.4
$RSub_{noRR}$	57.4	0.4	27.7	0.6	75.7	2.5
$SSub_{noRR}$	**55.8**	0.3	**24.4**	0.3	45.0	0.4
5%						
$WholeSet$	**27.1**	0.4	12.8	0.2	37.1	0.2
$RSub_{subOnly}$	32.1	0.2	14.9	0.1	**35.4**	0.2
$RSub_{noRR}$	29.1	0.2	12.6	0.1	70.6	3.2
$SSub_{noRR}$	28.5	0.2	**12.3**	0.1	35.8	0.2
10%						
$WholeSet$	**18.8**	0.3	**9.5**	0.1	34.7	0.1
$RSub_{subOnly}$	22.5	0.1	11.4	0.1	**32.4**	0.1
$RSub_{noRR}$	20.3	0.1	9.7	0.1	64.7	3.6
$SSub_{noRR}$	19.8	0.1	**9.5**	0.1	33.4	0.1

The first observation that can be made from table 3 is that $SSub_{noRR}$ consistently outperforms (or does about the same as) $RSub_{noRR}$, which validates our subset selection step (alg. 3). However, rather surprisingly, $RSub_{subOnly}$ can yield better performance than $WholeSet$ (on MNIST for 1% of labeled data, and systematically on COVTYPE): adding more unlabeled data actually harms the classification accuracy. There may be various reasons to this, the first one being that hyperparameters should be optimized separatly for each algorithm to get their best performance. In addition, for high-dimensional data without obvious clusters or low-dimensional representation, it is known that the inter-points distances tend to be all the same and meaningless (see e.g. (Beyer et al., 1999)). Thus, using a Gaussian kernel will force us to consider rather large neighborhoods, which prevents a sensible propagation of labels through the data during training. Nevertheless, a constatation that arises from those results is that $SSub_{noRR}$ never "breaks down", being always either the best or close to the best. It is able to take advantage of all the unlabeled data, while focussing the computations on a well chosen subset. The importance of the subset selection is made clear with COVTYPE, where choosing a random subset can be catastrophic: this is probably because the approximation made in eq. 8 is very poor for some of the points which are not in the subset, due to the low structure in the data.

Note that the goal here is not to obtain the best performance, but to compare the effectiveness of those algo-

rithms under the same experimental settings. Indeed, further refinements of the weighting function (see section 5.2) can greatly improve classification accuracy.

Additional experiments were performed to asses the superiority of our subset selection algorithm over random selection. In the following, unless specified otherwise, datasets come from the UCI MLR, and were preprocessed with standard normalization. The kernel bandwidth was approximately chosen to optimize the performance of $RSub_{noRR}$, and λ was arbitrarily set to 100. The experiments consist in taking as training set 67% of the available data, 10% of which are labeled, and using the subset approximation methods $RSub_{noRR}$ and $SSub_{noRR}$ with a subset of size 10% of the available unlabeled training data. The classification error is then computed on the rest of the data (test set), and averaged over 50 runs. On average, on the 8 datasets tested, $SSub_{noRR}$ always gives better performance. The improvement was *not* found to be statistically significative for the following datasets: Mushroom (8124 examples $\times$ 21 variables), Statlog Landsat Satellite (6435×36) and Nursery (12960×8). $SSub_{noRR}$ performs significantly better than $RSub_{noRR}$ (with a relative decrease in classification error from 4.5 to 12%) on: Image (2310×19), Isolet (7797×617), PenDigits (10992×16), SpamBase (4601×57) and the USPS dataset (9298×256, not from UCI). Overall, our experiments show that random selection can sometimes be efficient enough (especially with large low-dimensional datasets), but smart subset selection is to be preferred, since it (almost always) gives better and more stable results.

6 CONCLUSION

The first contribution of this paper is an extension of previously proposed non-parametric (graph-based) semi-supervised learning algorithms, that allows one to efficiently perform function induction (i.e. cheaply compute a prediction for a new example, in time $O(n)$ instead of $O(n^3)$). The extension is justified by the minimization of the same smoothness criterion used to obtain the original algorithms in the first place.

The second contribution is the use of this induction formula to define new optimization algorithms speeding up the training phase. Those new algorithms are based on using of a small subset of the unlabeled data, while still keeping information from the rest of the available samples. This subset can be heuristically chosen to improve classification performance over random selection. Such algorithms yield important reductions in computational and memory complexity and, combined with the induction formula, they give predictions close to the (expensive) transductive predictions.

References

Belkin, M., Matveeva, I., & Niyogi, P. (2004). Regularization and semi-supervised learning on large graphs. *COLT'2004*. Springer.

Belkin, M., & Niyogi, P. (2003). Using manifold structure for partially labeled classification. *Advances in Neural Information Processing Systems 15*. Cambridge, MA: MIT Press.

Beyer, K. S., Goldstein, J., Ramakrishnan, R., & Shaft, U. (1999). When is "nearest neighbor" meaningful? *Proceeding of the 7th International Conference on Database Theory* (pp. 217–235). Springer-Verlag.

Chapelle, O., Weston, J., & Scholkopf, B. (2003). Cluster kernels for semi-supervised learning. *Advances in Neural Information Processing Systems 15*. Cambridge, MA: MIT Press.

Kemp, C., Griffiths, T., Stromsten, S., & Tenembaum, J. (2004). Semi-supervised learning with trees. *Advances in Neural Information Processing Systems 16*. Cambridge, MA: MIT Press.

Nadaraya, E. (1964). On estimating regression. *Theory of Probability and its Applications*, *9*, 141–142.

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2002). On spectral clustering: analysis and an algorithm. *Advances in Neural Information Processing Systems 14*. Cambridge, MA: MIT Press.

Seeger, M. (2001). *Learning with labeled and unlabeled data* (Technical Report). Edinburgh University.

Szummer, M., & Jaakkola, T. (2002). Partially labeled classification with markov random walks. *Advances in Neural Information Processing Systems 14*. Cambridge, MA: MIT Press.

Watson, G. (1964). Smooth regression analysis. *Sankhya - The Indian Journal of Statistics*, *26*, 359–372.

Williams, C. K. I., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems 13* (pp. 682–688). Cambridge, MA: MIT Press.

Zhou, D., Bousquet, O., Navin Lal, T., Weston, J., & Schölkopf, B. (2004). Learning with local and global consistency. *Advances in Neural Information Processing Systems 16*. Cambridge, MA: MIT Press.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003a). Semi-supervised learning using gaussian fields and harmonic functions. *ICML'2003*.

Zhu, X., Lafferty, J., & Ghahramani, Z. (2003b). *Semi-supervised learning: From gaussian fields to gaussian processes* (Technical Report CMU-CS-03-175). CMU.

Structured Variational Inference Procedures and their Realizations

Dan Geiger*
Computer Science Department
Technion
Haifa, 36000, Israel
dang@cs.technion.ac.il

Christopher Meek
Microsoft Research
Microsoft Cooperation
Redmond, WA 98052, USA
meek@microsoft.com

Abstract

We describe and prove the convergence of several algorithms for approximate structured variational inference. We discuss the computation cost of these algorithms and describe their relationship to the mean-field and generalized-mean-field variational approaches and other structured variational methods.

1 Introduction

Graphical models are an important class of probabilistic models. Their graphical structure, whether directed, undirected, or mixed, provides an appealing description of the qualitative properties of the model. Furthermore, the modularity of the defined probability distribution allows one to define general algorithms, called inference algorithms, for computing marginal and conditional probabilities and allows one to easily incorporate prior knowledge. Inference algorithms are also useful for parameter learning for graphical models with missing data because the E-step of the EM algorithm can be computed using inference.

Although the inference problem is tractable for graphical models with small treewidth, the general inference problem is NP-hard (Cooper, 1990; Dagum and Luby, 1993) . In fact, for many graphical models of interest the treewidth is too large to allow efficient inference and one must use approximate or heuristic inference methods. In this paper, we examine the family of approaches that optimize the KL divergence between a distribution Q and the target distribution P where Q is constrained to be from some family of distributions for which inference is tractable.

* This work was partially done while the author visited Microsoft Research.

One of the nice properties of this family of approaches is that they provide a bound on marginal probabilities that are useful in model evaluation and learning. In particular, let us assume that we are given an intractable joint distribution $P(X)$ over a set of discrete variables X and our goal is to compute the marginal probability $P(Y = y)$ where $Y \subseteq X$. We let $H = X \setminus Y$. The quantity of interest is bounded by $\log P(Y = y) \geq -D(Q(H) \,\|\, P(Y = y, H))$ where $D(\cdot \,\|\, \cdot)$ denotes the KL divergence between two probability distributions. The quantity $-D(Q \,\|\, P)$ is often called the free-energy and denoted by $F(Q; P)$ where Q and P are possibly un-normalized distributions. The bound can be shown by the following argument:

$$\begin{aligned} -D(Q(H) \,\|\, P(Y = y, H)) &= -\sum_h Q(h) \log \frac{Q(h)}{P(y, h)} \\ &= \sum_h Q(h) \log P(y) - \sum_h Q(h) \log \frac{Q(h)}{P(h|y)} \\ &= \log P(y) - D(Q(H) \,\|\, P(H|Y = y)) \leq \log P(y). \end{aligned}$$

The final inequality follows from the fact that $D(Q(H) \,\|\, P(H|Y = y)) \geq 0$ with equality holding only if $Q(H) = P(H|Y = y)$. It is important to note that if Q is tractable then $D(Q(H) \,\|\, P(Y = y, H))$ can be effectively computed. The goal of approaches in this family is to find the $Q(H)$ that minimizes $D(Q(H) \,\|\, P(Y = y, H))$ (or maximizes the free-energy). Approaches in this family include the mean field, generalized mean field, and structured mean field approaches to variational inference. These methods differ with respect to the family of approximating distributions that can be used with the structural mean field approach subsuming the remaining approaches as special cases.

In this paper, we develop a set of structural variational methods inspired by the sequence of papers Saul and Jordan (1996), Ghahramani and Jordan (1997), Wiegerinck (2000) and Bishop and Winn (2003). We make several contributions with respect to this earlier

work. We provide a set of alternative structured variational methods and prove convergence of the alternatives with a novel simple proof technique. Our alternative algorithms differ in their computational profile with successive algorithms providing refined control over the computational cost of obtaining a variational approximation. We note that special cases of our final algorithm, called VIP$^\sharp$, were used in Jojic et al. (2004) for applying variational inference techniques to types of phylogenic models. For $N \times N$ grid-like models, algorithm VIP$^\sharp$ is $4N$ fold faster than algorithm VIP$^+$ and $12N$ folder faster than algorithm VIP, yielding a potential three orders of magnitude improvement in applications such as phylogeny and genetic linkage analyses.

2 Single Potential Update Algorithms

We denote distributions by $P(x)$ and $Q(x)$ and related un-normalized distributions by $\tilde{P}(x) \propto P(X)$ and $\tilde{Q}(x) \propto Q(x)$. Let X be a set of variables and x be an instantiation of these variables. Let $P(x) = \frac{1}{Z_P}\prod_i \Psi_i(d_i)$ where d_i is the projection of the instantiation x to the variables in $D_i \subseteq X$. The constant Z_P normalizes the product of potentials and the subsets $\{D_i\}_{i=1}^I$ are allowed to be overlapping. Note that we often suppress the arguments of a potential and of a distribution, using Φ_j instead of $\Phi_j(c_j)$ and P instead of $P(X)$.

Our goal is to find a distribution Q that minimizes the KL distance between Q and P. We further constrain Q to be of the form $Q(x) = \frac{1}{Z_Q}\prod_j \Phi_j(c_j)$ where Z_Q is a normalizing constant and where $C_1, \ldots, C_J$ are possibly overlapping subsets of X, which we call clusters. Finding an optimum Q, however, can be difficult. We set a more modest goal of finding a distribution Q which is a stationary point for the KL distance between Q and P, that is, $\nabla_\Phi D(Q \,\|\, P) = 0$ where $\Phi = \{\Phi_j\}_j$.

An algorithm, called VIP (for Variational Inference Procedure), that finds such a distribution Q is given in Figure 1. The algorithm uses the following indicator functions: $g_{kj} = 0$ if $C_k \cap C_j = \emptyset$ and 1 otherwise, and $f_{ij} = 0$ if $D_i \cap C_j = \emptyset$, and 1 otherwise. VIP relies at each step on an (inference) algorithm to compute some conditional probabilities from an unnormalized distribution $\tilde{Q}$ represented by a set of potentials $\Phi_j(c_j)$, $j = 1, \ldots, J$. This is accomplished by using *bucket elimination algorithm* or the *sum-product algorithm* described in (Dechter, 1999; Kschischang et al., 2001) as follows. To compute $Q(a|b)$ the algorithm first computes $\tilde{Q}(a, b)$ and then $\tilde{Q}(b)$. The conditional distribution of interest is the ratio of these two quantities because the normalizing constant cancels. It is important to note that for $\tilde{Q}(x) = \prod_j \Phi_j(c_j)$ the computation of these conditionals is not affected by multiplying any Φ_j by a constant α.

Algorithm VIP generalizes the mean field (MF) algorithm and the generalized mean field (GMF) algorithm (Xing et al. 2003,2004). The mean field algorithm is the special case of VIP in which each C_j contains a single variable. Similarly, the generalized mean field algorithm is the special case in which the C_j are disjoint subsets of variables. Note that if C_j are disjoint clusters, then the formula for γ_j in VIP simplifies to the GMF equations as follows (first term drops out):

$$\gamma_j(c_j) \leftarrow \sum_{\{i:f_{ij}=1\}} \sum_{D_i \setminus C_j} Q(d_i|c_j) \log \psi_i(d_i). \quad (2)$$

The term $Q(d_i|c_j)$ can be made more explicit when C_j are disjoint clusters. In particular, we partition the set $D_i \setminus C_j$ into $D_i^k = (D_i \setminus C_j) \cap C_k$ for $k = 1, \ldots, J$ where $k \neq j$. Note that $D_i^k = D_i \cap C_k$. Using this notation we have $Q(d_i|c_j) = \prod_k Q(d_j^k)$ where $Q(d_j^k) = 1$ whenever $D_i^k = \emptyset$. This factorization further simplifies the formula for γ_j in VIP as follows:

$$\gamma_j(c_j) \leftarrow \sum_{\{i:f_{ij}=1\}} \sum_{D_i^1} Q(d_i^1) \ldots \sum_{D_i^J} Q(d_i^J) \log \psi_i(d_i) \quad (3)$$

We note that this simplification is achieved automatically by the usage of bucket elimination for computing γ_j. The iterated sums in Eq. 3 are in fact the buckets formed by bucket elimination when C_j are disjoint.

Wiegerinck (2000) presents a less refined version of the update equation (Equation 1) and proves convergence to a stationary point of the KL distance between Q and P among all distributions Q of the given form using this update equation. We provide an alternative novel proof of convergence for our refined version of Wiegerinck's algorithm in Section 4 as a corollary to Theorem 1.

Equation 1 of VIP requires the computation of the quantities $Q(c_k|c_j)$ and $Q(d_i|c_j)$, and this is done in VIP using the bucket elimination algorithm. However, because there could be many indices k such that $C_k \cap C_j$ is not empty, and many indices i such that $D_i \cap C_j$ is not empty, these function calls are repeatedly applied independent of each other. However, these computations share many sub-computations, and it is therefore reasonable to add a data structure to facilitate a more efficient implementation for these function calls. In particular, it is possible to save computations if the sets $C_1, \ldots, C_J$ form a junction tree.

A set of clusters $C_1, \ldots, C_J$ forms a *junction tree* iff there exists a tree JT having one node, called C_j, for each subset of variables C_j, and for every two nodes C_i and C_j of JT, which are connected with a path in JT,

Algorithm VIP(Q, P)

Input: A set of potentials $\Psi_i(d_i)$ defining a probability distribution P via $P(x) = \frac{1}{Z_P}\prod_i \Psi_i(d_i)$ and a set of clusters C_j, $j = 1, \ldots, J$, with initial non-negative potentials $\Phi_j(c_j)$.

Output: A revised set of potentials $\Phi_j(c_j)$ defining a probability distribution Q via $Q(x) = \frac{1}{Z_Q}\prod_j \Phi_j(c_j)$ where Z_Q is a normalizing constant, such that Q is a stationary point of the KL distance $D(Q \,\|\, P)$.

Iterate over all clusters C_j until convergence

For every instantiation c_j of cluster C_j do:

$$\gamma_j(c_j) \leftarrow -\sum_{\{k:g_{kj}=1\}} \sum_{C_k \setminus C_j} Q(c_k|c_j) \log Q(c_k|c_j) + \sum_{\{i:f_{ij}=1\}} \sum_{D_i \setminus C_j} Q(d_i|c_j) \log \Psi_i(d_i) \quad (1)$$

using the sum-product algorithm on $\tilde{Q}(X) = \prod_i \Phi_i(C_i)$ to compute the quantities $Q(c_k|c_j)$ and $Q(d_i|c_j)$.

$\Phi_j(c_j) \leftarrow e^{\gamma_j(c_j)}$

Figure 1. The VIP algorithm

and for each node C_k on this path, $C_i \cap C_j \subseteq C_k$ holds. By a tree we mean an undirected graph, not necessarily connected, with no cycles. Note that this definition allows a junction tree to be a disconnected graph. When $C_1, \ldots, C_J$ form a junction tree, $Q(x)$ has the decomposable form $Q(x) = \prod_j \Phi_j(c_j) / \prod_e \Phi_e(s_e)$, where Φ_j are marginals on the subsets C_j of X, $j = 1, \ldots, J$, and where Φ_e are the marginals on intersections $S_e = C_i \cap C_j$, one for each two neighboring clusters in the junction tree (Jensen 1996).

The revised algorithm, which we call VIP$^+$, maintains a consistent junction tree JT for the distribution $Q(x)$. By consistency we mean that $\sum_{C_j \setminus C_k} \Phi_j = \sum_{C_k \setminus C_j} \Phi_k$ for every two clusters. In a consistent junction tree, each potential $\Phi_j(C_j)$ is proportional to $Q(C_j)$. There are two standard operations for junction trees: DISTRIBUTEEVIDENCE(Φ_j), and COLLECTEVIDENCE(Φ_j) (Jensen 1996). Algorithm VIP$^+$ uses the former. The procedure DISTRIBUTEEVIDENCE(Φ_j) accepts as input a consistent junction tree and a new cluster marginal Φ_j for C_j, and outputs a consistent junction tree, having the same clusters, where Φ_j is the (possibly unnormalized) marginal probability of Q on C_j, and where the conditional probability $Q(X|C_j)$ remains unchanged. Algorithm VIP$^+$ is given in Figure 2. This algorithm is identical to Wiegerink's algorithm except that the normalizing constant is not computed in each iteration. The fact that algorithm VIP$^+$ converges to a distribution Q which is a stationary point of the KL distance $D(Q \,\|\, P)$ is proved in Section 4 in Theorem 1.

Next we compare the computational benefit of VIP$^+$ versus VIP. The algorithms differ in two ways. First, VIP$^+$ makes the junction tree consistent with respect to the updated cluster. Second, VIP$^+$ uses junction tree inference to compute the quantities $Q(c_k|c_j)$ and $Q(d_i|c_j)$ whereas VIP uses the sum-product algorithm.

Most of the computation in both algorithms is directed towards computing conditional probabilities $Q(c_k|c_j)$ and $Q(d_i|c_j)$. We distinguish among these conditional probabilities as follows.

Definition: A conditional probability $Q(A|c_j)$ is *subsumed* by Q if the set of target variables A is a subset of some cluster C_k in Q (i.e., $A \setminus C_j \subseteq C_k$).

In the non-subsumed case, the set of target variables spans multiple clusters (i.e., $A \setminus C_j \not\subseteq C_k$). Clearly, all probabilities of the form $Q(C_k|c_j)$ are subsumed by Q.

The cost of running both the junction tree algorithm and the sum-product algorithm to compute a subsumed conditional probability $Q(A|c_j)$ is exponential in the treewidth of the model Q. The cost of the junction tree algorithm is typically twice the cost of the sum-product algorithm but, as we see below, this extra factor can be useful in reducing overall costs. For non-subsumed conditionals, both algorithms can cost upto a multiplicative factor of the size of the non-subsumed set. In the case of using junction trees, one can use the *variable propagation* algorithm in Jensen (1996) for each non-subsumed conditional.

The next example highlights the computational difference between VIP and VIP$^+$.

Example 1 *The target distribution P is a square grid of pairwise potentials (see Figure 3a) and the approximating family is defined by the set of columns in the*

Algorithm VIP$^+(Q, P)$

Input: A set of potentials $\Psi_i(d_i)$ defining a probability distribution P via $P(x) = \frac{1}{Z_P}\prod_i \Psi_i(d_i)$ and a set of clusters C_j, $j = 1, \ldots, J$, with initial potentials $\Phi_j(c_j)$ that form a consistent junction tree JT.

Output: A revised set of potentials $\Phi_j(c_j)$ defining a probability distribution Q via $Q(x) = \prod_j \Phi_j(c_j) / \prod_e \Phi_e(s_e)$, such that Q is a stationary point of the KL distance $D(Q \,\|\, P)$.

Note: *The potentials Φ_j are consistent un-normalized marginals encoding $\tilde{Q} \propto Q$. This fact is an invariant of the loop due to initialization and Step 2.*

Iterate over all clusters C_j until convergence

Step 1. For every instantiation c_j of cluster C_j do:

$$\gamma_j(c_j) \leftarrow - \sum_{\{k:g_{kj}=1\}} \sum_{C_k \setminus C_j} Q(c_k|c_j) \log Q(c_k|c_j) + \sum_{\{i:f_{ij}=1\}} \sum_{D_i \setminus C_j} Q(d_i|c_j) \log \Psi_i(d_i) \qquad (4)$$

where the quantities $Q(c_k|c_j)$ and $Q(d_i|c_j)$ are computed via the junction tree algorithm operating on JT.

$\Phi_j(c_j) \leftarrow e^{\gamma_j(c_j)}$

Step 2. Make JT consistent with respect to Φ_j: DISTRIBUTEEVIDENCE(Φ_j)

Figure 2. The VIP$^+$ algorithm

grid and denoted by Q_F (see Figure 3b) in which the clusters C_i ($i = 1, \ldots, 30$) correspond to edges.

Note that all conditionals required by the algorithms when optimizing Q_F are subsumed. In this example, for each c_j not on the boundary, there are six conditional probabilities that need to be computed. By using the junction tree algorithm all of these conditional probabilities can be computed with one call to DistributeEvidence whereas, when using the sum-product algorithm, each of these is computed separately. This yields a 3-fold speed up for VIP$^+$ with respect to VIP. For those c_j on a boundary, the speedup is a factor less than 3. As the size of the grid grows, a smaller fraction of the edges are on the boundary, and, thus, the speedup approaches a 3-fold speedup. For small grids, VIP$^+$ can be slower than VIP.

3 Multiple Potential Update Algorithm

In this section, we develop an algorithm to update multiple potentials at once to reduce the computational cost of optimizing the Q distribution. Algorithms VIP and VIP$^+$ do not assume any structure for Φ_j, namely, these algorithms hold tables Φ_j with an explicit entry for every instantiation of C_j. Since the computations $Q(c_k|c_j)$ and $Q(d_i|c_j)$ grow exponentially in the size of D_i and C_k, these algorithms become infeasible for large cliques or clusters. However, when structure is added to Φ_j, these algorithms can be modified to be more efficient by simultaneously updating this structure. In particular, one can use structure of the form,

$$\Phi_j(c_j) = \prod_{l=1}^{n_j} \Phi_{jl}(c_{jl}),$$

where the sets C_{jl} are possibly overlapping subsets of C_j, and c_{jl} is the projection of the instantiation c_j on the variables in C_{jl}. The potentials Φ_{jl} are assumed to be full tables and to form a junction tree JT_j.

Central to our development is a compatibility condition which allows us to simultaneously update the potentials.

Definition: A distribution Q with clusters C_j and subsets C_{jl} is *compatible* with a distribution P with sets D_i if for every D_i and C_j the set of indices $B_{ij} = \{l : D_i \cap C_j \subseteq C_{jl}\}$ is non-empty.

Our refined algorithm, VIP$^\sharp$, given in Figure 4, uses an indicator function $f_{ij}(l)$ which equals 0 when $D_i \cap C_j = \emptyset$, and when $D_i \cap C_j \neq \emptyset$, it equals 1 for a single fixed index $l \in B_{ij}$ and 0 for all other indices in B_{ij}. Our algorithm is closely related to the algorithm in Bishop and Winn (2003). As in Bishop and Winn (2003), we assume that the clusters of the approximating distribution are independent, that is, $Q(C_k|c_j) = Q(C_k)$. Our algorithm also generalizes the algorithm employed in (Jojic et al. 2004), which concentrates on specific models for phylogenetic analysis. We prove convergence of VIP$^\sharp$ in Section 4.

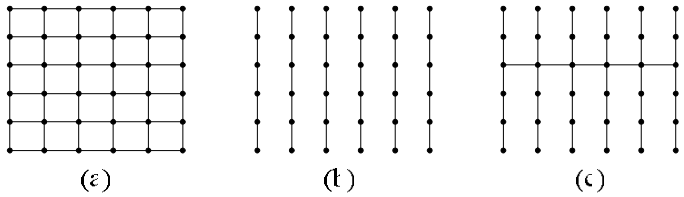


Figure 3. (a) Grid-like P distribution (b) factored structured distribution Q_F (c) connected distribution Q_C.

Example 2 *The target distribution P is a square grid of pairwise potentials (see Figure 3a) and the approximating family is a defined by the set of columns in the grid and denoted by Q_F (see Figure 3b) where C_i ($i = 1, \ldots, 6$) are columns of the grid.*

The approximating family with clusters defined by columns in this example satisfy the compatibility condition and the independence condition required by our algorithm.

Example 3 *The target distribution P is a square grid of pairwise potentials (see Figure 3a) and the approximating family is a defined by the set of columns in the grid and denoted by Q_C (see Figure 3c) where C_1 is the connected row of the grid and C_i ($i = 2, \ldots, 7$) are columns of the grid.*

The approximating family defined in Example 3 satisfies the compatibility condition but not the independence condition required by our algorithm. Note that, while the approximating family in Example 3 cannot be optimized using our refined algorithm below, it can be optimized using either VIP or VIP$^+$ in which, for instance, the C_i each contain a single edge.

We use Examples 1 and 2 to compare the benefits of VIP$^\sharp$ as compared to VIP$^+$. To analyze the difference between VIP$^\sharp$ and VIP$^+$ we need to analyze the number of times that one needs to call DistributeEvidence while computing conditional and marginal probabilities.

We begin by noting that the update Equation 5 in VIP$^\sharp$ takes advantage of the strong independence assumption to factor $Q(D_i|c_j)$, yielding a set of marginal probabilities that do not depend on c_j. Furthermore, the assumption of compatibility between P and Q implies that all the conditionals are subsumed. The factorization and compatibility conditions imply that each of these marginal probabilities can be obtained by lookup from the appropriate junction tree without calling DistributeEvidence. Therefore, we need no calls to DistributeEvidence in Step 1 and only one call to an inference algorithm to calibrate the junction tree associated with the potential being updated (Step 2).

In VIP$^+$, for each cluster C_j (edge) not in the boundary of the grid, there are twenty four conditionals that we need to compute, six for each of the four possible values for the cluster C_j. Every group of six conditionals can be updated with a single call to DistributeEvidence for a given c_j which gives four calls to the DistributeEvidence per cluster (edge). Again, as the size of the $N \times N$ grid grows, a smaller fraction of the edges are on the boundary, and, thus, the speedup approaches a 4N-fold speedup for VIP$^\sharp$ as compared to VIP$^+$, and 12N-fold as compared to VIP.

4 Proof of Convergence

In order to prove convergence of our algorithms, namely, that they converge to a stationary point of the KL distance between Q and P among all distributions Q of the given form, we examine properties of the KL distance between two distributions Q and P. Our proof technique is novel in that it uses properties of the KL distance rather than being based on Lagrangians (e.g., Wiegerinck 2000). The following lemmas furnish the needed properties of KL via basic algebra.

Lemma 1 *Let $P(x) = \frac{1}{Z_P} \prod_i \Psi_i(c_i)$ and $Q(x) = \frac{1}{Z_Q} \prod_j \Phi_j(d_j)$. Then,*

$$D(Q \,\|\, P) = \sum_{C_j} Q(c_j) \log \frac{Q(c_j)}{\Gamma_j(c_j)} + \log(Z_P) \qquad (6)$$

where $\Gamma_j(c_j) = e^{\gamma_j(c_j)}$ and where

$$\gamma_j(c_j) = -\sum_k \sum_{C_k \setminus C_j} Q(c_k|c_j) \log Q(c_k|c_j) + \sum_i \sum_{D_i \setminus C_j} Q(d_i|c_j) \log \Psi_i(d_i).$$

Proof: Recall that

$$D(Q \,\|\, P) = \sum_x Q(x) \log \frac{Q(x)}{P(x)} = -\left[H(Q) + E_Q[\log P(x)]\right] \qquad (7)$$

ALGORITHM VIP$^\sharp$ (Q,P)

Input: A set of disjoint clusters C_j, $j = 1, \ldots, J$ and a nested structure C_{jl} $(l = 1 \ldots, n_j)$ where $Q(c) \propto \prod_{j,l} \Phi_{jl}(c_{jl})$. A set of potentials $\Psi_i(d_i)$ defining a probability distribution P via $P(x) = \frac{1}{Z_P} \prod_i \Psi_i(d_i)$ such that the potentials are compatible with Q. A set of junction trees (JT_j) for each cluster C_j and a set of initial potentials $\Phi_j(c_j) = \prod_{l=1}^{n_j} \Phi_{jl}(c_{jl})$.
Output: A revised set of potentials $\Phi_{jl}(c_j)$ defining a probability distribution Q via $Q(x) = \prod_j Q_j(c_j)$ where $Q(c_j) \propto \prod_{l=1}^{n_j} \Phi_{jl}(c_{jl})$ such that Q is a stationary point of $D(Q \,\|\, P)$.

Iterate over all clusters C_j until convergence

Step 1. Compute messages for $l = 1, \ldots, n_j$:

For every instantiation c_{jl} of C_{jl} do:

$$\gamma_{jl}(c_{jl}) \leftarrow \sum_{\{i: f_{ij}(l)=l\}} \sum_{D_i^1} Q(d_i^1) \ldots \sum_{D_i^J} Q(d_i^J) \log \Psi_i(d_i) \tag{5}$$

where the quantities $Q(d_i^k)$ can be obtained by lookup in JT_k for Φ_k.

$$\Phi_{jl}(c_{jl}) \leftarrow e^{\gamma_{jl}(c_{jl})}$$

Note: *$\Phi_{jl}(c_{jl})$, $l = 1, \ldots, n_j$, implicitly encode the potential $\Phi_j(c_j)$, which is not being held explicitly anymore as in* VIP*, via $\Phi_j(c_j) = \prod_{l=1}^{n_j} \Phi_{jl}(c_{jl})$.* Recall that $D_i^k = D_i \cap C_k$.

Step 2. Make JT_j consistent with respect to Φ_j: DISTRIBUTEEVIDENCE(Φ_j)

Figure 4. The VIP$^\sharp$ algorithm

where $H(Q)$ denotes the entropy of $Q(x)$ and E_Q denotes expectation with respect to Q. The entropy term can be written as

$$H(Q) = -\sum_{C_j} Q(c_j) \log Q(c_j) - \sum_{C_j} Q(c_j) \sum_{X \setminus C_j} Q(x|c_j) \log Q(x|c_j)$$

where the first term is the entropy of $Q(C_j)$ and the second term is the conditional entropy of $Q(X|C_j)$. This well known form of $H(Q)$ is derived by splitting summation over X into summation over C_j and over $X \setminus C_j$, and using the fact that $\sum_{X \setminus C_j} Q(x|c_j) = 1$. By splitting the sum over $X \setminus C_j$, this entropy term is further rewritten as

$$H(Q) = -\sum_{C_j} Q(c_j) \log Q(c_j) - \sum_{C_j} Q(c_j) \sum_k \sum_{C_k \setminus C_j} Q(c_k|c_j) \log Q(c_k|c_j).$$

The second term of Eq. 7 is similarly written as

$$\begin{aligned} E_Q[\log P(x)] = & \\ &= \sum_i \sum_{C_j} Q(c_j) \sum_{X \setminus C_j} Q(x|c_j) \log \Psi_i(d_i) - \log(Z_P) \\ &= \sum_{C_j} Q(c_j) \sum_i \sum_{D_i \setminus C_j} Q(d_i|c_j) \log \Psi_i(d_i) - \log(Z_P) \end{aligned}$$

Hence Eq. 7 is rewritten as

$$D(Q \,\|\, P) = \textstyle\sum_{C_j} Q(c_j) \log Q(c_j) - \sum_{C_j} Q(c_j) \left[-\sum_k \sum_{C_k \setminus C_j} Q(c_k|c_j) \log Q(c_k|c_j) + \sum_i \sum_{D_i \setminus C_j} Q(d_i|c_j) \log \Psi_i(d_i) \right] + \log(Z_P)$$

Denoting the bracketed term by $\gamma_j(c_j)$, and letting $\Gamma_j(c_j) = e^{\gamma_j(c_j)}$, we get

$$D(Q \,\|\, P) = \sum_{C_j} Q(c_j) \log \frac{Q(c_j)}{\Gamma_j(c_j)} + \log(Z_P). \qquad \diamond$$

Note that $\Gamma_j(c_j)$ in Eq. 6 does not depend on $Q(c_j)$ and is a function of $Q(x)$ only through the conditional distribution of $X \setminus C_j$ given C_j (via $Q(c_k|c_j)$). Eq. 6 states that the KL distance between $Q(x)$ and $P(x)$ is equal, up to an additive constant, to the KL distance between $Q(c_j)$ and an un-normalized potential $\Gamma_j(c_j)$. This interesting result generalizes a similar equation for a special case derived in (Jojic et al, 2004).

The next lemma provides a variant of a well known property of KL. Recall that for every two probability distributions $Q(x)$ and $P(x)$, the KL distance

$D(Q(x) \,\|\, P(x)) \geq 0$ and equality holds if and only if $Q(x) = P(x)$ (Cover and Thomas 1991; Theorem 2.6.3). A similar result holds also for un-normalized probability distributions.

Lemma 2 *Let $\tilde{Q}(x)$ and $\tilde{P}(x)$ be non-negative functions such that $\sum_x \tilde{P}(x) = Z_P > 0$, and let*

$$\hat{Q}(x) = \min_{\{\tilde{Q} | \sum_x \tilde{Q}(x) = Z_Q\}} D(\tilde{Q}(x) \,\|\, \tilde{P}(x))$$

where Z_Q is a positive constant. Then $\hat{Q}(x) = \frac{Z_Q}{Z_P} P(x)$.

Proof. We observe that

$$D(\tilde{Q}(x) \,\|\, \tilde{P}(x)) = Z_Q \cdot D(\tfrac{\tilde{Q}(x)}{Z_Q} \,\|\, \tfrac{\tilde{P}(x)}{Z_P}) + \log \frac{Z_Q}{Z_P}$$

which implies, using the cited result about normalized distributions, that the minimum is obtained when $\frac{\tilde{Q}(x)}{Z_Q} = \frac{\tilde{P}(x)}{Z_P}$, yielding the desired claim. $\diamond$

Theorem 1 (Convergence of VIP$^+$) *Algorithm VIP$^+$ converges to a stationary point of the KL distance between Q and P among all distributions Q of the form $Q(x) = \frac{1}{Z_Q} \prod_j \Phi_j(c_j)$.*

Proof. We need to show that at the start of each iteration of VIP$^+$ the function Q defined by the revised potentials $\Phi_j(c_j)$ is closer to P in KL distance than Q at the start of the previous iteration. We rewrite the KL distance $D(Q \,\|\, P)$ using Eq. 6, as justified by Lemma 1. Using the given form of Q, we have

$$Q(c_j) = \frac{1}{Z_Q} \left[\sum_{X \setminus C_j} \prod_{k \neq j} \Phi_k(c_k) \right] \Phi_j(c_j). \qquad (8)$$

We denote the bracketed coefficient of $\Phi_j(c_j)$ by B and note that it is constant in the sense that it does not depend on the quantity Φ_j being optimized. We now use Eq. 8 to rewrite Eq. 6 as

$$D(Q \,\|\, P) = \frac{B}{Z_Q} \left[\sum_{C_j} \Phi_j(c_j) \log \frac{\Phi_j(c_j)}{\Gamma_j(c_j)} \right] + \log \frac{B Z_P}{Z_Q}. \qquad (9)$$

Recall that $\Gamma_j(c_j) = e^{\gamma_j(c_j)}$ does not depend on $\Phi_j(c_j)$ since it only depends on the conditional probability $Q(X|c_j)$. Hence, Lemma 2 states that the (global) minimum wrt Φ_j is achieved when $\Phi_j(c_j)$ is set to be proportional to $\Gamma_j(c_j)$. It is possible to set $\Phi_j(c_j)$ to be proportional to $\Gamma_j(c_j)$, as done in Step 1 of VIP$^+$, because $\Phi_j(c_j)$ is a full potential. The proportionality constant does not matter because if Φ_j is multiplied by α, and the arbitrary constraining constant Z_Q is also multiplied by α, these influences cancel in Eq. 9. For simplicity, in the algorithm, we use $\alpha = 1$ and therefore $\Phi_j(c_j) \leftarrow e^{\gamma_j(c_j)}$. Algorithm VIP$^+$ computes $\Phi_j(c_j)$ according to this formula and hence decreases the KL distance in each iteration by improving $\Phi_j(c_j)$ while holding all other cluster potentials fixed. Since the KL distance is lower bounded by zero, VIP$^+$ converges to a stationary point.

It remains to show that at the start of each iteration, the quantities $Q(c_k|c_j)$ and $Q(d_i|c_j)$ can be computed correctly from the junction tree for the un-normalized distribution $\tilde{Q}$ of the current normalized distribution Q. At each iteration, Q is computed up to some implicit normalizing factor, say α, so that $\tilde{Q}(x) = \alpha Q(x)$. The procedure DISTRIBUTEEVIDENCE(Φ_j) is based on the following update scheme. Starting with Φ_j, every neighboring cluster node C_k in the junction tree, representing the cluster potential $\Phi_k(c_k)$, is updated via

$$\Phi_k^{new}(c_k) \leftarrow \Phi_k(c_k) \frac{\Phi_k^{new}(s_{jk})}{\Phi_k(s_{jk})}$$

where s_{jk} is the instantiation for the separator $S_{jk} = C_j \cap C_k$ consistent with the instantiation c_k, and then the cluster neighbors of the neighboring clusters are updated similarly, until all clusters have been updated (Jensen, 1996). In each step, any normalizing constant implicitly appears 4 times in this update equation, and it cancels out regardless of its value. Hence, the quantities $Q(c_k|c_j)$ and $Q(d_i|c_j)$ are updated correctly from the un-normalized distribution $\tilde{Q}$. $\diamond$

Corollary 1 (Convergence of VIP) *Algorithm VIP converges to a stationary point for the KL distance between Q and P among all distributions Q of the form $Q(x) = \frac{1}{Z_Q} \prod_j \Phi_j(c_j)$.*

Proof. VIP convergence follows from Theorem 1 because the update equations for the two algorithms, Equations 1 and 4, are identical; the two algorithms only differ in the method by which conditional probabilities are computed. $\diamond$

Theorem 2 (Convergence of VIP$^\sharp$) *Algorithm VIP$^\sharp$ converges to a stationary point of the KL distance between Q and P among all distributions Q of the form $Q(x) = \frac{1}{Z_Q} \prod_{jl} \Phi_{jl}(c_{jl})$ where Q and P are compatible.*

Proof: We analyze Equation 4 in light of the assumptions made in VIP$^\sharp$. The first term in Equation 4 is constant with respect to c_j and, thus, does not effect the update and can be dropped. Next, the fact that the clusters C_j of Q are independent means that

$Q(D_i) = \prod_k Q(D_i^k)$ where $D_i^k = D_i \cap C_k$. This factorization implies that $Q(D_i|c_j) = \prod_{k \neq j} Q(D_i^k)$ which in turn implies that $F_i(c) = \sum_{D_i \setminus C_j} Q(d_i|c_j) \log \Psi_i(d_i)$ equals $\sum_{D_i^1} Q(d_i^1) \ldots \sum_{D_i^J} Q(d_i^J) \log \Psi_i(d_i)$. By the assumption of compatibility, each $F_i(c)$ is a function of c_{jl} (i.e., after summing out all variables in $D_i \setminus C_j$) and, thus, can be put into the potential $\Phi_{jl}(c_{jl})$. Every element in the second sum of Equation 4 is put into some potential and $\Gamma_j(c_j)$ from Theorem 1 is equal to $\prod_l e^{\gamma_{jl}(c_{jl})}$. Therefore, by updating, for all l, $\Phi_{jl}(c_{jl}) \propto \Gamma_{jl}(c_{jl})$ is equivalent to updating Φ_j in Theorem 1 and, thus, the algorithm converges.$\diamond$

Acknowledgments

We thank Nir Friedman, David Heckerman, and Nebojsa Jojic for various discussions on the subject of variational techniques.

References

Bishop, C. & Winn, J. (2003). Structured variational distributions in VIBES. In *Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics.

Cooper, G. (1990). Probabilistic inference using belief networks is NP-hard. *Artificial Intelligence*, *42*, 393–405.

Cover, T. M. & Thomas, J. A. (1991). *Elements of Information Theory*. Wiley.

Dagum, P. & Luby, M. (1993). Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, *60*(1), 141–153.

Dechter, R. (1999). Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, *113*(1-2), 41–85.

Ghahramani, Z. & Jordan, M. I. (1997). Factorial hidden Markov models. *Machine Learning*, *29*, 245–273.

Jensen, F. V. (1996). *An Introduction to Bayesian Networks*. UCL Press.

Jojic, V., Jojic, N., Meek, C., Geiger, D.,Siepel, A., Haussler, D., & Heckerman, D. (2004). Efficient approximations for learning phylogenetic HMM models from data. *Bioinformatics*, *20*, 161–168.

Kschischang, F. R., Frey, B. J., & Loeliger, H.-A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, *47*(2).

Saul, L. & Jordan, M. I. (1996). Exploiting tractable substructures in intractable networks. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press.

Wiegerinck, W. (2000). Variational approximations between mean field theory and the junction tree algorithm. In *Uncertainty in Artificial Intelligence*. Morgan Kaufmann.

Xing, E. P., Jordan, M. I., & Russell, S. (2003). A generalized mean field algorithm for variational inference in exponential families. In *Uncertainty in Artificial Intelligence*. Morgan Kaufmann.

Xing, E. P., Jordan, M. I., & Russell, S. (2004). Graph partition strategies for generalized mean field inference. In *Uncertainty in Artificial Intelligence*. Morgan Kaufmann.

Kernel Constrained Covariance for Dependence Measurement

Arthur Gretton*, Alexander Smola†, Olivier Bousquet*, Ralf Herbrich‡, Andrei Belitski*, Mark Augath*, Yusuke Murayama*, Jon Pauls*, Bernhard Schölkopf*, & Nikos Logothetis*
first.last@tuebingen.mpg.de, Alex.Smola@anu.edu.au, rherb@microsoft.com
* MPI for Biological Cybernetics, Tübingen, Germany
† NICTA, Canberra, Australia; ‡ Microsoft Research, Cambridge, UK

Abstract

We discuss reproducing kernel Hilbert space (RKHS)-based measures of statistical dependence, with emphasis on constrained covariance (COCO), a novel criterion to test dependence of random variables. We show that COCO is a test for independence if and only if the associated RKHSs are universal. That said, *no* independence test exists that can distinguish dependent and independent random variables in all circumstances. Dependent random variables can result in a COCO which is arbitrarily close to zero when the source densities are highly non-smooth. All current kernel-based independence tests share this behaviour. We demonstrate exponential convergence between the population and empirical COCO. Finally, we use COCO as a measure of joint neural activity between voxels in MRI recordings of the macaque monkey, and compare the results to the mutual information and the correlation. We also show the effect of removing breathing artefacts from the MRI recording.

1 Introduction

Tests to determine the dependence or independence of random variables are well established in statistical analysis. Some approaches require density estimation as an intermediate step ([13] is a classic study); while others assume a parametric model of how the variables were obtained from independent random variables, as in blind source separation [12].

In this paper we propose a non-parametric independence criterion, which relies on the fact that the random variables[1] x, y are independent if and only if

$$\mathbf{E}_{\mathsf{x}}[f(\mathsf{x})]\mathbf{E}_{\mathsf{y}}[g(\mathsf{y})] = \mathbf{E}_{\mathsf{x},\mathsf{y}}[f(\mathsf{x})g(\mathsf{y})]. \tag{1.1}$$

for bounded, continuous functions f, g (see for instance [14, 19]). The proposed criterion works by maximising the discrepancy between the empirical estimates of the LHS and RHS of (1.1) over pre-specified function classes $f \in \mathcal{F}$ and $g \in \mathcal{G}$, and comparing the discrepancy to the amount of deviation that can be expected from the fact that we are dealing with empirical estimates rather than expectations. We call our criterion the constrained covariance (COCO).[2]

The results presented here build on recent work published on the subject of kernel based dependence measures. In particular, the canonical correlation between functions in a reproducing kernel Hilbert space (KCC), defined in [1] for a variety of kernels and in [15] for splines, can be used as a test of independence. Indeed, in the case of Gaussian kernels, Bach and Jordan show the KCC to be zero if and only if its two arguments are statistically independent. In Section 3, we characterise *all* reproducing kernel Hilbert spaces (RKHSs) for which this property holds (both for COCO and KCC): these are required to be *universal* (the RKHS must be dense in the space of continuous functions [22]). Specifically, the Gaussian and Laplace kernels are universal, as are many exponential-based kernels; polynomial kernels, however, are not universal.

We next demonstrate in Section 4 that for a fixed-size, finite sample of *dependent* random variables, there exists no test that can reliably detect that the random variables are dependent. To clarify how this might affect our criterion, we prove that the population COCO can be made arbitrarily small when certain smoothness assumptions on the density are violated, which makes it difficult to detect dependence on the basis of a finite sample. This is also true of other related kernel dependence measures, including the kernel mutual information (KMI) in [9], and kernel generalised variance (KGV) in [1], both of which were shown in [9] to be upper bounds near independence on the Parzen window estimate of the mutual information. Thus, as in all dependence tests, any inference made is subject to certain assumptions about the underlying generative process - the present work describes these assumptions explicitly for the first time, in the case of kernel-based tests.

[1] We write random variables *sans serif*.

[2] In [9], this was called the kernel covariance (KC).

Next, we give two bounds, based on Rademacher averages, which describe *exponential* convergence between the population and empirical COCO. The first assures us that the population COCO is small when the empirical COCO is small; the second shows that the population COCO is large when the empirical COCO is large (both statements apply with high probability). These results are very interesting, in that they illustrate a broader phenomenon: slow learning rates do not occur in dependence testing, even though they are unavoidable in regression and classification [6, Ch. 7]. This might appear surprising in the specific case of COCO, since this criterion is optimised in the course of kernelised PLS regression (assuming a kernelised output space: see the discussion in [2]). Another important consequence of the bounds is that *any* dependence between the random variables will be detected rapidly *as the sample size increases*, even though perfect dependence detection is impossible for fixed sample size.

Finally, we describe a neuroscience application where our method can be used. A number of groups (e.g. [3]) have begun examining the interactions between neural systems using fMRI in humans. The recent study of BOLD fMRI in the macaque monkey using high field (4.7T & 7T) scanners [17, 16] has resulted in substantial increases in spatial and temporal resolution, when measuring brain activity patterns resulting from various stimuli. In Section 6, we apply COCO to these high resolution data so as to detect dependence between BOLD responses within the visual cortex. In using COCO to detect regions of high dependence, we follow [8], who maximise a kernel-based dependence measure (in their case, the KGV) as a means of variable selection. We also investigate how the measured dependence changes with the removal of breathing artefacts, which is feasible due to the high temporal resolution of our measurements.

2 Definitions and Background

Before presenting our main results, we begin our discussion with some relevant definitions and background theory, covering both classical independence criteria and RKHSs.[3] Let $(\Omega, \mathcal{A}, \mathbf{P}_{\mathsf{x},\mathsf{y}})$ be a probability space. Consider random variables $\mathsf{x} : (\Omega, \mathcal{A}) \to (U, \mathcal{U})$ and $\mathsf{y} : (\Omega, \mathcal{A}) \to (V, \mathcal{V})$, where U and V are complete metric spaces, and $\mathcal{U}$ and $\mathcal{V}$ their respective Borel σ-algebras. The covariance between x and y is defined as follows.

Definition 1 (Covariance). *The covariance of two random variables* x, y *is given as*

$$\mathrm{cov}(\mathsf{x}, \mathsf{y}) := \mathbf{E}_{\mathsf{x},\mathsf{y}}[\mathsf{x}\mathsf{y}] - \mathbf{E}_{\mathsf{x}}[\mathsf{x}]\mathbf{E}_{\mathsf{y}}[\mathsf{y}]. \tag{2.1}$$

For our purposes, the notion of independence of random variables is best expressed using the following characterisation:

[3] This exposition is necessarily dense: see [14] and [21] for more detail.

Theorem 1 (Independence [14]). *The random variables* x *and* y *are independent if and only if* $\mathrm{cov}(f(\mathsf{x}), g(\mathsf{y})) = 0$ *for each pair* (f, g) *of bounded, continuous functions.*

This theorem suggests the following definition as an independence test.

Definition 2 (Constrained covariance). Given function classes $\mathcal{F}, \mathcal{G}$ containing subspaces $F \in \mathcal{F}$ and $G \in \mathcal{G}$, we define the *constrained covariance* as

$$\mathrm{COCO}(\mathbf{P}_{\mathsf{x},\mathsf{y}}; F, G) := \sup_{f \in F, g \in G} [\mathrm{cov}(f(\mathsf{x}), g(\mathsf{y}))]. \tag{2.2}$$

(if F and G are unit balls in their respective spaces, then this is just the norm of the covariance operator mapping $\mathcal{G}$ to $\mathcal{F}$: see [8] and references therein). Given n independent observations $\boldsymbol{z} := ((x_1, y_1), \ldots, (x_n, y_n)) \subset (\mathcal{X} \times \mathcal{Y})^n$, its empirical estimate is defined as

$$\mathrm{COCO}_{\mathrm{emp}}(\boldsymbol{z}; F, G) := \sup_{f \in F, g \in G} \left[\frac{1}{n} \sum_{i=1}^{n} f(x_i) g(y_i) - \frac{1}{n^2} \sum_{i=1}^{n} f(x_i) \sum_{j=1}^{n} g(y_j) \right].$$

It follows from Theorem 1 that if $\mathcal{F}, \mathcal{G}$ are the sets of continuous functions bounded by 1 we have $\mathrm{COCO}(\mathbf{P}_{\mathsf{x},\mathsf{y}}; \mathcal{F}, \mathcal{G}) = 0$ if and only if x and y are independent.[4] In other words, COCO and $\mathrm{COCO}_{\mathrm{emp}}$ are criteria which can be tested *directly* without the need for an intermediate density estimator (in general, the distributions may not even have densities). It is also clear, however, that unless F, G are restricted in further ways, $\mathrm{COCO}_{\mathrm{emp}}$ will always be large, due to the rich choice of functions available. A *non-trivial dependence measure* is thus obtained using function classes that do not give an everywhere-zero empirical average, yet which still guarantee that COCO is zero if and only if its arguments are independent. A tradeoff between the restrictiveness of these function classes and the convergence of $\mathrm{COCO}_{\mathrm{emp}}$ to COCO can be accomplished using standard tools from uniform convergence theory (see Section 5).

It turns out (Section 3) that unit-radius balls in universal reproducing kernel Hilbert spaces constitute function classes that yield non-trivial dependence estimates. To demonstrate this, we will use certain properties of these spaces [20]. A reproducing kernel Hilbert space is a Hilbert space $\mathcal{F}$ for which at each $x \in \mathcal{X}$, the point evaluation functional, $\delta_x : \mathcal{F} \to \mathbb{R}$, which maps $f \in \mathcal{F}$ to $f(x) \in \mathbb{R}$, is continuous. To each reproducing kernel Hilbert space, there corresponds a unique positive definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (the reproducing kernel), which constitutes the inner product on this space: this is guaranteed by the Moore-Aronszajn theorem.

[4] Here we set $F = \mathcal{F}$ and $G = \mathcal{G}$.

In RKHSs the representer theorem [21] holds, stating that the solution of an optimisation problem, dependent only on the function evaluations on a set of observations and on RKHS norms, lies in the span of the kernel functions evaluated on the observations. This property is next used to specify an easily computed expression for $\mathrm{COCO}_{\mathrm{emp}}(\boldsymbol{z}; F, G)$ where F and G are respectively unit balls in the reproducing kernel Hilbert spaces $\mathcal{F}$ and $\mathcal{G}$. The proof may be found in [9].

Lemma 1 (Value of $\mathrm{COCO}_{\mathrm{emp}}(\boldsymbol{z}; F, G)$). *Denote by $\mathcal{F}, \mathcal{G}$ RKHSs on the domains $\mathcal{X}$ and $\mathcal{Y}$ respectively and let F, G be the unit balls in the corresponding RKHS. Then*

$$\mathrm{COCO}_{\mathrm{emp}}(\boldsymbol{z}; F, G) = \frac{1}{n}\sqrt{\|\bar{K}^f \bar{K}^g\|_2} \tag{2.3}$$

where $\bar{K}^f$ is the matrix obtained by the projection $\bar{K}^f = PK^fP$ with projection operator $P_{ij} = \delta_{ij} - \frac{1}{n}$ and Gram matrix $K^f_{ij} = k_f(x_i, x_j)$. $\bar{K}^g$ is defined by analogy using the kernel of $\mathcal{G}$ (which might be different from that of $\mathcal{F}$).

A second theorem which will be crucial in our proofs is Mercer's theorem, which provides a decomposition of the kernel into eigenfunctions and eigenvalues.

Theorem 3 (Mercer's theorem). *Let $k(\cdot,\cdot) \in L_\infty(\mathcal{X}^2)$ be a symmetric real valued function with an associated positive definite integral operator with normalised orthogonal eigenfunctions $\varphi_p \in L_2(\mathcal{X})$, sorted such that the associated eigenvalues $\tilde{k}_p$ do not increase. Then for almost all $x_i \in \mathcal{X}$ and $x_j \in \mathcal{X}$, the series*

$$k(x_i, x_j) := \sum_{p=1}^{\infty} \tilde{k}_p \varphi_p(x_i)\varphi_p(x_j)$$

converges absolutely and uniformly. In addition, the sum $\sum_{i=1}^{p} |\tilde{k}_i|$ converges as $p \to \infty$.

Finally, we give kernel-dependent decay rates for the coefficients used to expand functions in $\mathcal{F}$ in terms of the set of basis functions $\{\varphi_i(\cdot)\}$ from Mercer's theorem.

Lemma 4 (Rate of decay of expansion coefficients). *Let $f \in \mathcal{F}$, where $f(x) := \sum_{i=1}^{\infty} \tilde{f}_i \varphi_i(x)$. Then as long as $(\tilde{k}_i)^{-1}$ increases super-linearly with i, $(|\tilde{f}_i|) \in \ell_1$ and there exists an $l_0 \in \mathbb{N}$ such that for all $\epsilon > 0$ and all $l > l_0$, $|\tilde{f}_l| < \epsilon$.*

Proof. This holds since for any $f \in \mathcal{F}$, $\|f\|^2_{\mathcal{F}} := \sum_{i=1}^{\infty} \tilde{f}_i^2 \left(\tilde{k}_i\right)^{-1} < \infty$. □

The super-linearity requirement in Lemma 4 is satisfied by many kernels, including the Gaussian (for which the $((\tilde{k}_m)^{-1})$ increase as $\exp(m^2)$); see [21]. We assume hereafter that our kernel satisfies the requirements of Lemma 4.

3 A Test for Independence

We now characterise the class of kernels for which COCO is a non-trivial test of dependence. The main result is given in Theorem 6, in which we demonstrate that COCO constitutes such a test when $\mathcal{F}$ and $\mathcal{G}$ are RKHSs with a universal kernel [22].

Definition 5 (Universal kernel). A *continuous kernel $k(\cdot,\cdot)$* on a compact metric space $(\mathcal{X}, d)$ is called universal if and only if the RKHS $\mathcal{F}$ induced by the kernel is dense in $C(\mathcal{X})$ with respect to the topology induced by the infinity norm $\|f - g\|_\infty$.

For instance, [22] shows the following two kernels are universal on compact subsets of $\mathbb{R}^d$:

$$k(x, x') = \exp\left(-\lambda\|x - x'\|^2\right) \text{ and}$$
$$k(x, x') = \exp\left(-\lambda\|x - x'\|\right) \text{ for } \lambda > 0.$$

We now state our main result for this section.

Theorem 6 (COCO is only zero at independence for universal kernels). *Denote by $\mathcal{F}, \mathcal{G}$ RKHSs with universal kernels k_f, k_g on the compact domains $\mathcal{X}$ and $\mathcal{Y}$ respectively and let F, G be the unit balls in the corresponding RKHSs. We assume without loss of generality that $\|f\|_\infty \leq 1$ and $\|g\|_\infty \leq 1$ for all $f \in \mathcal{F}$ and $g \in \mathcal{G}$. Then $\mathrm{COCO}(\mathsf{P}_{\mathsf{x},\mathsf{y}}; F, G) = 0$ if and only if x, y are independent.*

Proof. It is clear that $\mathrm{COCO}(\mathsf{P}_{\mathsf{x},\mathsf{y}}; F, G)$ is zero if x and y are independent. We prove the converse by contradiction, using the starting assumptions $\mathrm{COCO}(\mathsf{P}_{\mathsf{x},\mathsf{y}}; B(\mathcal{X}), B(\mathcal{Y})) = c$ for some $c > 0$ (here $B(\mathcal{X})$ denotes the subset of $C(\mathcal{X})$ of continuous functions bounded by 1 in the $L_\infty(\mathcal{X})$, and $B(\mathcal{Y})$ is defined in an analogous manner) and $\mathrm{COCO}(\mathsf{P}_{\mathsf{x},\mathsf{y}}; F, G) = 0$. There exist two sequences of functions $f_n \in C(\mathcal{X})$ and $g_n \in C(\mathcal{Y})$, satisfying $\|f_n\|_\infty \leq 1, \|g_n\|_\infty \leq 1$, for which

$$\lim_{n\to\infty} \mathrm{cov}(f_n(\mathsf{x}), g_n(\mathsf{y})) = c.$$

More to the point, there exists an n^* for which $\mathrm{cov}(f_{n^*}, (\mathsf{x})g_{n^*}(\mathsf{y})) \geq c/2$. We know that $\mathcal{F}$ and $\mathcal{G}$ are respectively dense in $C(\mathcal{X})$ and $C(\mathcal{Y})$: this means that for all $1/3 > \epsilon > 0$, we can find some $f^* \in \mathcal{F}$ (and an analogous $g^* \in \mathcal{G}$) satisfying $\|f^* - f_{n^*}\|_\infty < \epsilon = \frac{c}{24}$. Writing as $\tilde{f}(\mathsf{x}) := f^*(\mathsf{x}) - f_{n^*}(\mathsf{x}) + f_{n^*}(\mathsf{x})$ (with an analogous $\tilde{g}(\mathsf{y})$ definition), we obtain

$$\begin{aligned}
&\mathrm{cov}(f^*(\mathsf{x}), g^*(\mathsf{y})) \\
&= \mathsf{E}_{\mathsf{x},\mathsf{y}}\left[\tilde{f}(\mathsf{x})\tilde{g}(\mathsf{y})\right] - \mathsf{E}_{\mathsf{x}}(\tilde{f}(\mathsf{x}))\mathsf{E}_{\mathsf{y}}(\tilde{g}(\mathsf{y})) \\
&\geq \mathrm{cov}(f_{n^*}(\mathsf{x}), g_{n^*}(\mathsf{y})) - 2\epsilon\left|\mathsf{E}_{\mathsf{x}}\left(f_{n^*}(\mathsf{x})\right)\right| \\
&\qquad - 2\epsilon\left|\mathsf{E}_{\mathsf{y}}\left(g_{n^*}(\mathsf{y})\right)\right| - 2\epsilon^2 \\
&\geq \frac{c}{2} - 6\frac{c}{24} = \frac{c}{4} > 0.
\end{aligned}$$

This contradicts the assumption that $\mathrm{cov}(f^*(\mathsf{x}), g^*(\mathsf{y})) = 0$, and completes the proof. □

The kernel dependence tests (COCO, KMI, KGV, and KCC) are generalised in [9, 1] to a greater number of random variables, providing tests of pairwise independence.

4 Limitations of Independence Tests

4.1 General independence tests

In this section, we illustrate with a simple example that for a finite sample, there exists no test of independence which can reliably (i.e. with high probability) distinguish dependence from independence. This discussion is intended as a complement to the next section, where we explicitly construct dependent random variables which are difficult for the empirical COCO to distinguish from independence. We illustrate the case where $\mathcal{X}$ is countable, but our reasoning applies equally to continuous spaces.

We begin with some notation. Consider a set $\mathcal{P}$ of probability distributions $\mathbf{P}_{\mathsf{x}}$, where x contains m entries. The set $\mathcal{P}$ is split into two subsets: $\mathcal{P}_i$ contains distributions $\mathbf{P}_{\mathsf{x}}^{(i)}$ of mutually independent random variables $\mathbf{P}_{\mathsf{x}}^{(i)} = \prod_{j=1}^{m} \mathbf{P}_{\mathsf{x}_j}$, and $\mathcal{P}_d$ contains distributions $\mathbf{P}_{\mathsf{x}}^{(d)}$ of dependent random variables. We next introduce a test $\Delta(\boldsymbol{x})$, which takes a data set[5] $\boldsymbol{x} \sim \mathbf{P}_{\mathsf{x}}{}^{n}$, and returns

$$\Delta(\boldsymbol{x}) = 1 \ : \ \boldsymbol{x} \sim \mathbf{P}_{\mathsf{x}^n}^{(d)}, \qquad \Delta(\boldsymbol{x}) = 0 \ : \ \boldsymbol{x} \sim \mathbf{P}_{\mathsf{x}^n}^{(i)}$$

Given that the test sees only a finite sample, it cannot determine with complete certainty whether the data are drawn from $\mathbf{P}_{\mathsf{x}^n}^{(d)}$ or $\mathbf{P}_{\mathsf{x}^n}^{(i)}$. We call Δ an α-test when

$$\sup_{\mathbf{P}_{\mathsf{x}}^{(i)} \in \mathcal{P}_i} \mathbf{E}_{\boldsymbol{x} \sim \mathbf{P}_{\mathsf{x}^n}^{(i)}} (\Delta(\boldsymbol{x}) = 1) \leq \alpha;$$

in other words α upper bounds the probability of a Type I error. Our theorem is as follows:

Theorem 7 (Universal limit on dependence tests). *For any α-test, some fixed $n \in \mathbb{N}$, and any $1 - \alpha > \epsilon > 0$, there exists $\mathbf{P}_{\mathsf{x}} \notin \mathcal{P}_i$ such that*

$$\mathbf{P}_{\boldsymbol{x} \sim \mathbf{P}_{\mathsf{x}}{}^{n}} (\Delta(\boldsymbol{x}) = 0) \geq 1 - \alpha - \epsilon;$$

in other words, a dependence test with a low Type I error can have a severe Type II error.

Proof. We introduce a distribution $\mathbf{P}_{\mathsf{x}}^{(\gamma)} := \gamma \mathbf{P}_{\mathsf{x}}^{(i)} + (1-\gamma)\mathbf{P}_{\mathsf{x}}^{(d)}$, where $0 \leq \gamma < 1$. Clearly, random variables drawn from $\mathbf{P}_{\mathsf{x}}^{(\gamma)}$ are dependent. The probability of a Type II error for this mixture is then

$$\begin{aligned}
&\mathbf{P}_{\boldsymbol{x} \sim \mathbf{P}_{\mathsf{x}^n}^{(\gamma)}} (\Delta(\boldsymbol{x}) = 0) && \overset{(a)}{=} \sum_{\boldsymbol{x}} \mathbf{P}_{\mathsf{x}^n}^{(\gamma)}(\boldsymbol{x}) \mathbb{I}_{\Delta(\boldsymbol{x})=0} \\
&= \sum_{\boldsymbol{x}} \prod_{k=1}^{n} \mathbf{P}_{\mathsf{x}}^{(\gamma)}(\mathbf{x}_k) \mathbb{I}_{\Delta(\boldsymbol{x})=0} && = \sum_{\boldsymbol{x}} \prod_{k=1}^{n} \gamma \mathbf{P}_{\mathsf{x}}^{(i)}(\mathbf{x}_k) \mathbb{I}_{\Delta(\boldsymbol{x})=0} \\
&= \gamma^n \mathbf{P}_{\boldsymbol{x} \sim \mathbf{P}_{\mathsf{x}^n}^{(i)}} (\Delta(\boldsymbol{x}) = 0) && = \gamma^n (1 - \alpha)
\end{aligned}$$

where the sum following (a) is over all possible draws of $\boldsymbol{x}$ from $\mathbf{P}_{\mathsf{x}^n}^{(\gamma)}$, and $\mathbb{I}_A$ is the indicator function for event A. Taking γ very close to 1 (i.e. making the dependent distribution very unlikely in the mixture) proves the theorem. □

4.2 Kernel independence tests

We prove the existence of a dependent probability distribution for which COCO is small, but with a large covariance between certain functions in $\mathcal{F}$ and $\mathcal{G}$; we then demonstrate that this also holds for the KCC, KMI, and KGV. Although the population COCO is *not* zero for this density, its small size will make this dependence hard to detect unless a large data sample is available. We illustrate this phenomenon by specifying a particular joint density $\mathbf{f}_{\mathsf{x},\mathsf{y}}$ (with distribution $\mathbf{P}_{\mathsf{x},\mathsf{y}}$) chosen such that $\mathrm{cov}(\varphi_l(\mathsf{x}), \varphi_l(\mathsf{y}))$ is large for some large l (meaning x, y have a non-trivial dependence), but $\mathrm{COCO}(\mathbf{P}_{\mathsf{x},\mathsf{y}}; F, G)$ is small. The intuition behind our argument is made clear by re-writing COCO for RKHSs as

$$\mathrm{COCO}\left(\mathbf{P}_{\mathsf{x},\mathsf{y}}; F, G\right) = \sup_{f \in \mathcal{F},\, g \in \mathcal{G}} \frac{\mathrm{cov}\left(f(\mathsf{x}), g(\mathsf{y})\right)}{\|f\|_{\mathcal{F}} \|g\|_{\mathcal{G}}}. \quad (4.1)$$

This will obviously be small when the RKHS norms in the denominator are much larger than the covariance in the numerator: we will see that this motivates our choice of density. More specifically, high order eigenfunctions of the kernel[6] ($\varphi_l(x)$ and $\varphi_l(y)$ for large l) have large RKHS norms, a fact widely exploited in regression as a roughness penalty [21]. Thus, if the high order eigenfunctions are prominent in $\mathbf{f}_{\mathsf{x},\mathsf{y}}$ (i.e., for highly non-smooth densities), we expect COCO to be small even when there exists an l for which $\mathrm{cov}(\varphi_l(\mathsf{x}), \varphi_l(\mathsf{y}))$ is large.[7]

Theorem 8 (Dependent random variables can have small COCO). *There exists a density $\mathbf{f}_{\mathsf{x},\mathsf{y}}$ for which $\mathrm{cov}\left(\varphi_l(\mathsf{x}), \varphi_l(\mathsf{y})\right) \geq \beta - \epsilon$ for non-trivial β and arbitrarily small $\epsilon > 0$, yet for which $\mathrm{COCO}(\mathbf{P}_{\mathsf{x},\mathsf{y}}; F, G) < \gamma$ for an arbitrarily small $\gamma > 0$.*

Proof. The proof is a sketch only: further detail is given in [10]. We begin by constructing a density for which $\mathrm{cov}\left(\varphi_l(\mathsf{x}), \varphi_l(\mathsf{y})\right) \geq \beta - \epsilon$. This is written

$$\mathbf{f}_{\mathsf{x},\mathsf{y}}(x, y) = \alpha_l + \beta \varphi_l(x) \varphi_l(y) \quad (4.2)$$

where $\mathbf{f}_{\mathsf{x},\mathsf{y}}(x, y) \geq 0$ and $\int \mathbf{f}_{\mathsf{x},\mathsf{y}}(x, y) dx dy = 1$. The first constraint requires $\alpha_l - \beta \min_{x,y}(\varphi_l(x)\varphi_l(y)) \geq 0$,

[5] We denote by $\boldsymbol{x} \sim \mathbf{P}_{\mathsf{x}}{}^{n}$ the drawing of n i.i.d. samples $\boldsymbol{x} := (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ from $\mathbf{P}_{\mathsf{x}}$.

[6] See Theorem 3 for a definition of the eigenfunctions. Note that the kernels in $\mathcal{F}$ and $\mathcal{G}$ may not be identical, and the eigenfunctions $\varphi_i(x)$ and $\varphi_j(y)$ might therefore be different. We use the *arguments* of the eigenfunctions to distinguish between them, since this is unambiguous and avoids messy notation.

[7] This reasoning can be extended to motivate kernel choice for the detection of particular dependencies, although this is beyond the scope of the present study. Note also that an alternative Parzen-window based interpretation of kernel choice is given in [9].

which can be satisfied as long as the $\varphi_l(x)$ and $\varphi_l(y)$ are absolutely bounded.[8] The second constraint affects the covariance between kernel eigenfunctions,

$$\begin{aligned}\widetilde{C}_{i,j} &= \operatorname{cov}(\varphi_i(x), \varphi_j(y)) \\ &:= \mathbf{E}_{\mathsf{x},\mathsf{y}}(\varphi_i(\mathsf{x})\varphi_j(\mathsf{y})) - \mathbf{E}_{\mathsf{x}}(\varphi_i(\mathsf{x}))\mathbf{E}_{\mathsf{y}}(\varphi_j(\mathsf{y})).\end{aligned} \tag{4.3}$$

Indeed, this constraint causes $\widetilde{\mathbf{C}}$ to have i, jth entries

$$\widetilde{C}_{i\neq l, j\neq l} := \epsilon_{ij}, \qquad \widetilde{C}_{l,l} := \beta + \epsilon_{ll}, \tag{4.4}$$

where ϵ_{ij} denotes a quantity with absolute value arbitrarily small for large enough l (the proof requires some tedious algebra, but is not difficult: notably, it makes use of the decay result in Lemma 4).

We next expand the functions f and g which define COCO (i.e. elements of the respective RKHSs at which the supremum is attained) as $f(x) = \sum_{i=1}^{\infty} \tilde{f}_i \varphi_i(x)$ and $g(y) = \sum_{j=1}^{\infty} \tilde{g}_j \varphi_j(y)$ (the expansion coefficients are written as vectors $\tilde{\mathbf{f}}, \tilde{\mathbf{g}}$). Using these expansions, the numerator of (4.1) becomes $\operatorname{cov}(f(x), g(y)) = \tilde{\mathbf{f}}^\top \widetilde{\mathbf{C}} \tilde{\mathbf{g}}$, and

$$\begin{aligned}\tilde{\mathbf{f}}^\top \widetilde{\mathbf{C}} \tilde{\mathbf{g}} &\leq \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} |\tilde{f}_i|\,|\tilde{g}_j|\,\epsilon + |\tilde{f}_l|\,|\tilde{g}_l|\,\beta \\ &= \|\tilde{\mathbf{f}}\|_1 \|\tilde{\mathbf{g}}\|_1 \epsilon + |\tilde{f}_l|\,|\tilde{g}_l|\,\beta,\end{aligned}$$

where we replace all entries in $\widetilde{\mathbf{C}}$ with their expressions in (4.4), and ϵ is the ϵ_{ij} with largest absolute value. Lemma 4 ensures that $\|\tilde{\mathbf{f}}\|_1$ and $\|\tilde{\mathbf{g}}\|_1$ both converge. In the case of the remaining term $|\tilde{f}_l|\,|\tilde{g}_l|\,\beta$, we divide through by the norms in the denominator of COCO to get

$$\begin{aligned}&|\tilde{f}_l|\,|\tilde{g}_l|\,\beta \left(\sum_{i=1}^{\infty} \tilde{f}_i^2 \left(\tilde{k}_i^f\right)^{-1}\right)^{-\frac{1}{2}} \left(\sum_{j=1}^{\infty} \tilde{f}_j^2 \left(\tilde{k}_j^g\right)^{-1}\right)^{-\frac{1}{2}} \\ &\leq \beta\sqrt{\tilde{k}_l^f \tilde{k}_l^g},\end{aligned}$$

and the right hand side approaches zero as $l \to \infty$ thanks to Theorem 3.[9] □

We now address how the KCC [1] has the same limitation, being upper bounded by a constant multiple of COCO. The KCC is defined as

$$\begin{aligned}&\mathcal{J}_\kappa\left(\mathbf{P}_{\mathsf{x},\mathsf{y}}; \mathcal{F}, \mathcal{G}\right) \\ &:= \sup_{f\in\mathcal{F}, g\in\mathcal{G}} \frac{\operatorname{cov}\left(f(\mathsf{x}), g(\mathsf{y})\right)}{\sqrt{\operatorname{var}\left(f\left(\mathsf{x}\right)\right) + \kappa\|f\|_{\mathcal{F}}^2}\sqrt{\operatorname{var}\left(g\left(\mathsf{y}\right)\right) + \kappa\|g\|_{\mathcal{G}}^2}} \\ &\leq \kappa^{-1}\|f^*\|_{\mathcal{F}}^{-1}\|g^*\|_{\mathcal{G}}^{-1}\operatorname{cov}\left(f^*(\mathsf{x}), g^*(\mathsf{y})\right) \\ &\leq \kappa^{-1}\mathrm{COCO}\left(\mathbf{P}_{\mathsf{x},\mathsf{y}}; F, G\right),\end{aligned}$$

where f^*, g^* attain the supremum in the first line, and we assume f and g to be bounded.

Finally, we demonstrate that the KMI [9] and KGV [1], which are respectively extensions to COCO and the KCC, have the same property. This follows since the KMI can be written as $-\frac{1}{2}\log(\prod_{i=1}^{n}(1-\rho_i^2))$, where $|\rho_i|$ are upper bounded by COCO, and the KGV as $-\frac{1}{2}\log(\prod_{i=1}^{n}(1-\gamma_i^2))$, where the $|\gamma_i|$ are upper bounded by the KCC. Small COCO will therefore cause small KMI, and small KCC will cause small KGV.

5 Bounds

We give two convergence bounds in this section. The first (and simplest) guarantees small population COCO when the empirical COCO is small; the second, which has a more involved derivation, guarantees that if the empirical COCO is large, then the population COCO is also large. The proofs are given in sketch form only; rigorous derivations are provided in [10]. A consequence of these bounds is that the empirical COCO converges to the population COCO at speed $1/\sqrt{n}$. This means that if we define the independence test $\Delta(\boldsymbol{z})$ (Section 4.1) as the indicator that COCO is larger than a term of the form $C\sqrt{\log(1/\alpha)/n}$ with C a constant, then $\Delta(\boldsymbol{z})$ is an α-test with type II error upper bounded by a term approaching zero as $1/\sqrt{n}$.

Our first bound makes use of the following theorem, which applies to U-statistics of the kind we encounter in calculating covariances.

Theorem 9 (Positive deviation bound for one sample U-statistics [11, p. 25]). *Consider a collection of n i.i.d. random variables $(\mathsf{z}_1, \ldots, \mathsf{z}_n)$. We define the U-statistic*

$$\mathsf{u} := \frac{1}{n(n-1)\ldots(n-r+1)} \sum_{\mathbf{i}_r^n} h_{i_1,\ldots,i_r}(\mathsf{z}_{i_1}, \ldots, \mathsf{z}_{i_r}),$$

where the index set $\mathbf{i}_r^n$ is the set of all r-tuples drawn without replacement from $\{1, \ldots, n\}$, and the function h is called the kernel *of the U-statistic. If $a \leq h \leq b$,*

$$\mathbf{P}_{\mathsf{u}}(\mathsf{u} - \mathbf{E}_{\mathsf{u}}(\mathsf{u}) \geq t) \leq \exp\left(\frac{-2t^2\lceil n/r\rceil}{(b-a)^2}\right).$$

We now state the bound.

Theorem 10 (Upper bound on population COCO). *Assume that functions in F and G are*

[8] This condition is not satisfied for all Mercer kernels: see [21, Exercise 2.24]. The assumption holds in most everyday cases we encounter (e.g. the Fourier basis), however, so it is reasonable in this context.

[9] On the basis of this proof, we might suppose that using an RBF kernel with small width (and thus with a slow decay of the coefficients $\tilde{k}_i^f$ and $\tilde{k}_j^f$) would make COCO larger when dependence takes the form (4.2) above, with high order eigenfunctions $\varphi_l(x), \varphi_l(y)$. While this is true, the empirical estimate of COCO will become inaccurate if the spacing between samples significantly exceeds the kernel width: thus, there is a practical limit on how small we can make the kernel.

bounded a.s. by 1. Then for $n > 1$ and all $\delta > 0$, with probability at least $1 - \delta$,

$$\sup_{f\in F, g\in G} \mathrm{cov}(f(\mathsf{x}), g(\mathsf{y})) \leq \sup_{f\in F, g\in G} \widehat{\mathrm{cov}}(f(x), g(y)) + \Lambda.$$

where $\Lambda = \sqrt{\frac{2\log(2/\delta)}{n(\sqrt{2}-1)^2}}$, and we denote the empirical covariance based on the sample $\boldsymbol{z}$ as

$$\widehat{\mathrm{cov}}(f(x), g(y)) := \frac{1}{n(n-1)} \sum_{i\neq j} f_i g_j - \frac{1}{n}\sum_{i=1}^{n} f_i g_i,$$

where $f_i := f(x_i)$ and $g_j := g(y_j)$.

Proof. First, we rearrange

$$\sup_{f\in F, g\in G} \mathrm{cov}(f(\mathsf{x}), g(\mathsf{y})) - \sup_{f\in F, g\in G} \widehat{\mathrm{cov}}(f(x), g(y)) \leq \sup_{f\in F, g\in G} \left(\mathrm{cov}(f(\mathsf{x}), g(\mathsf{y})) - \widehat{\mathrm{cov}}(f(x), g(y))\right).$$

We can therefore ignore the suprema, and treat only the random variables $\mathsf{f} := f(\mathsf{x})$ and $\mathsf{g} := g(\mathsf{y})$. To complete the bound, we make the split

$$\mathbf{P}_{\mathsf{f}^n,\mathsf{g}^n}\left(\mathrm{cov}(\mathsf{f},\mathsf{g}) - \widehat{\mathrm{cov}}(f,g) \geq t\right) \leq \mathbf{P}_{\mathsf{f}^n,\mathsf{g}^n}\left(-\frac{1}{n}\sum_{i=1}^{n}\mathsf{f}_i\mathsf{g}_i + \mathbf{E}_{\mathsf{f},\mathsf{g}}(\mathsf{fg}) \geq (1-\alpha)t\right) + \mathbf{P}_{\mathsf{f}^n,\mathsf{g}^n}\left(\frac{1}{n(n-1)}\sum_{i\neq j}\mathsf{f}_i\mathsf{g}_j - \mathbf{E}_{\mathsf{f}}(\mathsf{f})\mathbf{E}_{\mathsf{g}}(\mathsf{g}) \geq \alpha t\right)$$

The first term is bounded straightforwardly using Hoeffding's inequality [11]. To bound the second term in the sum, we define the random vector $\mathsf{z}_i := (\mathsf{f}_i, \mathsf{g}_i)$, and the kernel $h_{i,j}(\mathsf{z}_i, \mathsf{z}_j) := \mathsf{f}_i\mathsf{g}_j$. It is clear that Theorem 9 then applies. We complete the proof by setting $\alpha = 2 - \sqrt{2}$. □

A lower bound on the population COCO is harder to compute, since we have to deal with the suprema.

Theorem 11 (Lower bound on population COCO). *Assume functions in F and G are bounded a.s. by 1, and that the functions $k_f(x,x) \leq 1$ and $k_g(y,y) \leq 1$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Then for $n > 1$ and all $\delta > 0$, with probability at least $1 - \delta$,*

$$\sup_{f\in F, g\in G} \widehat{\mathrm{cov}}(f,g) \leq \sup_{f\in F, g\in G} \mathrm{cov}(f,g) + \frac{134}{\sqrt{n}} + \sqrt{\frac{18\log 2/\delta}{n}}.$$

Proof. We begin with a rearrangement of the suprema;

$$\sup_{f\in F, g\in G} \widehat{\mathrm{cov}}(f(x), g(y)) - \sup_{f\in F, g\in G} \mathrm{cov}_{\mathsf{x},\mathsf{y}}(f(\mathsf{x}), g(\mathsf{y})) \leq \sup_{f\in F, g\in G}\left(\mathbf{E}_{\mathsf{x},\mathsf{y}} f(\mathsf{x})g(\mathsf{y}) - \frac{1}{n}\sum_{i=1}^{n} f(x_i)g(y_i)\right) + \sup_{f\in F, g\in G}\left(\frac{1}{n(n-1)}\sum_{i\neq j} f(x_i)g(y_j) - \mathbf{E}_{\mathsf{x}} f(\mathsf{x})\mathbf{E}_{\mathsf{y}} g(\mathsf{y})\right)$$

The first term is bounded using McDiarmid [18] and symmetrisation in the usual way. In the case of the second term, we begin with McDiarmid to get

$$\mathbf{P}_{\mathsf{x}^n,\mathsf{y}^n}\left(\sup_{f\in F, g\in G}\left(\frac{1}{n(n-1)}\sum_{i\neq j} f_i g_j - \mathbf{E}_{\mathsf{x}} f \mathbf{E}_{\mathsf{y}} g\right) \geq \underbrace{\mathbf{E}_{\mathsf{x}^n,\mathsf{y}^n}\sup_{f\in F, g\in G}\left[\frac{1}{n(n-1)}\sum_{i\neq j} f_i g_j - \mathbf{E}_{\mathsf{x}} f \mathbf{E}_{\mathsf{y}} g\right]}_{(a)} + t\right) \leq e^{\frac{-nt^2}{8}}.$$

We cannot symmetrise this expression directly: instead, we first apply the Hoeffding decomposition and then decouple, following [5]. This yields an upper bound on the expectation (a) that we can symmetrise. We do not go into detail, but the idea is to replace certain of the random variables by independent copies. After decoupling and symmetrisation, we obtain

$$\begin{aligned}(\mathrm{a}) \leq 32\,\mathbf{E}\sup_{f,g}\Bigg(&\frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{i\neq j}^{n}\sigma_i\sigma'_j\Big(f(\mathsf{x}_i)g(\mathsf{y}'_j) \\ &- f(\mathsf{x}_i)g(\mathsf{y}''_j) - f(\mathsf{x}''_i)g(\mathsf{y}'_j) + f(\mathsf{x}'_i)g(\mathsf{y}''_j)\Big)\Bigg) \\ &+\frac{4}{n}\mathbf{E}\sup_{f,g}\sum_{i=1}^{n}(\sigma_i f(\mathsf{x}_i)g(\mathsf{y}'_i)) = (b),\end{aligned}$$

where the σ_i are Rademacher random variables that take values in $\{-1, 1\}$ with equal probability, σ'_i are independent copies of σ_i, x'_i, x''_i are independent copies of x_i, and y'_i, y''_i are independent copies of y_i. To conclude the proof, it turns out that we do not need to explicitly deal with these additional copies: instead, we apply a simple additional bound (see [10]) to get

$$\begin{aligned}(b) \leq &\frac{128}{n(n-1)}\mathbf{E}_{\mathsf{x}^n,\mathsf{y}^n}\sqrt{\sum_{i\neq j} k_f(\mathsf{x}_i,\mathsf{x}_i)k_g(\mathsf{y}_i,\mathsf{y}_i)} \\ &+\frac{4}{n}\mathbf{E}_{\mathsf{x}^n,\mathsf{y}'^n}\sqrt{\sum_{i=1}^{n} k_f(\mathsf{x}_i,\mathsf{x}_i)k_g(\mathsf{y}'_i,\mathsf{y}'_i)},\end{aligned}$$

and then substitute $k_f(x,x) \leq 1$ and $k_g(y,y) \leq 1$. □

6 Experiments and discussion

We previously applied COCO in the context of independent component analysis (ICA) [9], where it performed similarly to the kernel canonical correlation [1]. Thus, COCO has been established in practice as a useful test of *independence*. In the present study, we give preliminary results obtained when using COCO to determine regions that show *high* dependence in

the macaque primary visual cortex. We are able to monitor neural activity in three different ways: (a) electrophysiology only with large electrode arrays, (b) fMRI only, and (c) combined electrode and fMRI measurements. The present section deals only with dependence measures on the fMRI signals, but we are currently expanding this analysis to cover a broader range of acquisition techniques [17], so as to compare the dependence found for these different types of measurement. Our fMRI readings were taken using a 4.7T scanner with a sampling frequency of 4Hz and a 96mmx96mm field of view (FOV), with resolution 256x256 and 0.5mm slice thickness, in accordance with the procedure in [16]. The stimulus used was a clip from "Star Wars", which was chosen so as to excite a broad range of activity within the visual cortex. Dependence was investigated for voxels in the primary visual area (V1) for a total of 250 voxels, and the signal duration during stimulus was 250 seconds (1000 samples). The results obtained are an aggregate over 35 such experiments.

The observed fMRI signals were contaminated with a breathing component. Since the macaque monkey was under general anaesthetic during data acquisition, breathing was mechanically assisted, and had a constant frequency of approximately 0.4Hz. We modelled this breathing as being of constant amplitude and linearly superposed on the haemodynamic response. This model is motivated by the narrowness of the spectral peaks at the breathing frequency and harmonics, which suggests that any amplitude modulation of the breathing signal is of very low frequency, and can be assumed effectively constant. Thus, while we could not directly recover the true breathing contamination at each voxel, we were able to use the decrease in the spectral peak at the breathing frequency, averaged across all voxels, as a measure of success in removing the breathing artefacts. Harmonics at integer multiples of the fundamental frequency were modelled in the same way. The exact frequency of the breathing signal was found by averaging the spectrum over all voxels, and the phase at each voxel was chosen to maximise the projection in the time domain of the breathing sinusoid. Only voxels near large blood vessels were contaminated by the breathing signal, and thus a threshold test was applied to the spectrum at each voxel, to test whether a substantial breathing component was visible. Where breathing was present, a sinusoid of corresponding frequency and phase was projected out in the time domain (thus also removing the associated frequency domain sidelobes caused by finite signal duration). The same procedure was used to remove the first two harmonics. We did not band-pass filter the signal to remove the breathing, as this would have eliminated a greater portion of the spectral components due to the haemodynamic activity.

As dependence measures between pairs of voxels, we applied cross correlation between voxels, the mutual information (MI) (computed using the method in [4]), and COCO (using RKHSs with Gaussian kernels). The variation in dependence between voxels was studied with all three methods, as a function of average distance between voxels (in other words, we grouped together all pairs of voxels an equal distance from each other; we then clustered these pairs so as to draw together voxel pairs with similar distances). The regions of interest (ROI) were constrained to be convex sets so as to avoid incorrect distance estimates. Specifically, Euclidean distances at the image level may significantly differ from the actual axonic distances connecting neural sites. We subtracted a baseline dependence from each of the dependence measures, which was obtained by averaging the dependence between the V1 region and a test region of the brain, the latter being unrelated to visual processing. The dependence amplitudes were then divided by the standard deviation in the average dependence between V1 and this test region. Results are plotted in Figure 6.1.

Comparing the dependence measures before and after breathing removal shows significant effects of respiratory artefacts on the high order[10] dependence vs. distance curves (COCO and MI): this finding suggests extreme caution for studies in humans, in which respiration-induced signals cannot easily be modelled due to low temporal sampling rates, as well as variable respiration frequency and amplitude. Prior to breathing removal, COCO and the MI overestimate the dependence between voxels (the breathing artefacts being a source of considerable similarity), as does the correlation, though to a lesser extent. This can be explained by phase shifts between the breathing contamination observed at different voxels, which reduce the correlation but have less effect on more general measures of dependence. The high order dependence curves also flatten out at about 5mm once breathing is removed, but continue to decay with distance when breathing is present. By contrast, the correlation prior to breathing removal is constant (to within observational uncertainty) after about 2mm; following breathing removal, however, the point at which it flattens out is more difficult to determine. Finally, compared with the MI, COCO at short distances is a larger multiple of the standard deviation in test region dependence, which might make COCO a more reliable measure of such short range dependencies. On the other hand, both the MI and the correlation fall to a baseline level of activity greater than that in the test region, which COCO does not detect. Additional experiments on a greater number of subjects and stimuli will be used to verify these observations.

Further work will focus on the construction of dependence-distance functions, using for instance the well developed mathematical framework of flat brain-

[10] The correlation takes into account only second order dependence, whereas COCO and the MI can detect dependence of any order.

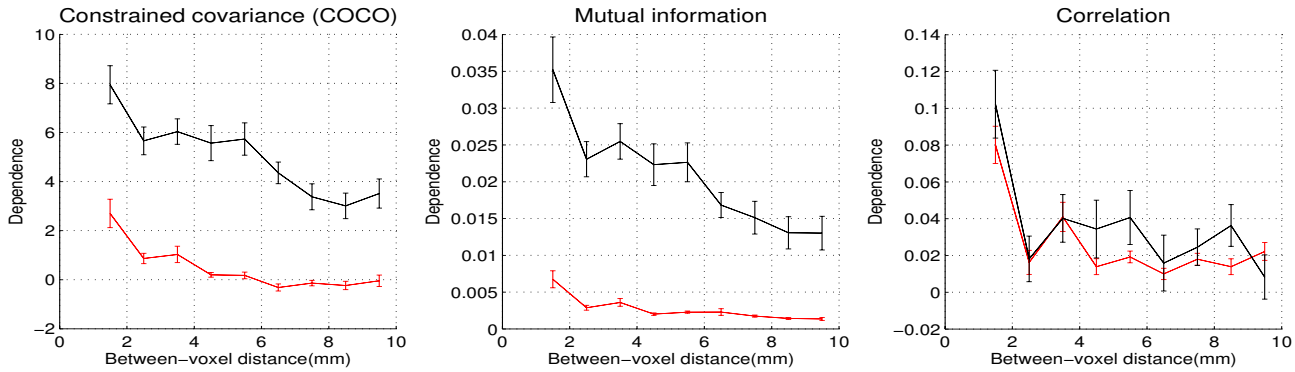


Figure 6.1: Results before (black) and after (red) breathing removal, for COCO, the mutual information (MI), and the correlation. The offset and scaling of the dependence is described in the main body of the text. The horizontal axis displays the average distance between voxel pairs.

map generation [7]. Dependence tests that take into account the fact that the signals are not i.i.d. will also be compared with the present approaches. In addition, it is of interest to develop kernel-based dependence measures for non-i.i.d. time series.

References

[1] F. Bach and M. Jordan. Kernel independent component analysis. *JMLR*, 3:1–48, 2002.

[2] G.H. Bakır, A. Gretton, M. Franz, and B. Schölkopf. Multivariate regression with stiefel constraints. Technical Report 101, Max Planck Institute for Biological Cybernetics, 2004.

[3] C. Buchel and K. Friston. Modulation of connectivity in visual pathways by attention: cortical interactions evaluated with structural equation modelling and fMRI. *Cerebral Cortex*, 7:768–778, 1997.

[4] G. A. Darbellay and I. Vajda. Estimation of the information by an adaptive partitioning of the observation space. *IEEE Transactions on Information Theory*, 45(4):1315–1321, 1999.

[5] V. de la Peña and E. Giné. *Decoupling: from Dependence to Independence*. Springer, New York, 1999.

[6] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*, volume 31 of *Applications of mathematics*. Springer, New York, 1996.

[7] D. Van Essen, H. Drury, S. Joshi, and M. Miller. Functional and structural mapping of human cerebral cortex: solutions are in the surfaces. *Proceedings of Nat. Acad. of Sciences of the USA*, 95:788–795, 1998.

[8] K. Fukumizu, F. R. Bach, and M. I. Jordan. Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *JMLR*, 5:73–99, 2004.

[9] A. Gretton, R. Herbrich, and A. Smola. The kernel mutual information. Technical report, MPI for Biological Cybernetics, 2003.

[10] A. Gretton, A. Smola, O. Bousquet, and R. Herbrich. Behaviour and convergence of the constrained covariance. Technical report, MPI for Biological Cybernetics, 2004.

[11] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[12] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley and Sons, New York, 2001.

[13] Yu. I. Ingster. An asymptotically minimax test of the hypothesis of independence. *J. Soviet Math.*, 44:466–476, 1989.

[14] J. Jacod and P. Protter. *Probability Essentials*. Springer, New York, 2000.

[15] S. E. Leurgans, R. A. Moyeed, and B. W. Silverman. Canonical correlation analysis when the data are curves. *J. R. Stat. Soc. B*, 55(3):725–740, 1993.

[16] N. Logothetis, H. Guggenberger, and J. Pauls S. Peled. Functional imaging of the monkey brain. *Nature Neuroscience*, 2:555–562, 1999.

[17] N. Logothetis, J. Pauls, M. Augath, T. Trinath, and A. Oeltermann. Neurophysiological investigation of the basis of the fMRI signal. *Nature*, 412:150–157, 2001.

[18] C. McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, pages 148–188, 1989. Cambridge University Press.

[19] A. Rényi. On measures of dependence. *Acta Math. Acad. Sci. Hungar.*, 10:441–451, 1959.

[20] S. Saitoh. *Theory of Reproducing Kernels and its Applications*. Longman Scientific and Technical, Harlow, UK, 1988.

[21] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT press, Cambridge, MA, 2002.

[22] I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *JMLR*, 2, 2001.

Semisupervised alignment of manifolds

Jihun Ham
Department of Electrical and Systems Engineering,
University of Pennsylvania,
Philadelphia, PA 19104

Daniel D. Lee
Department of Electrical and Systems Engineering,
University of Pennsylvania,
Philadelphia, PA 19104

Lawrence K. Saul
Department of Computer and Information Science,
University of Pennsylvania,
Philadelphia, PA 19104

Abstract

In this paper, we study a family of semisupervised learning algorithms for "aligning" different data sets that are characterized by the same underlying manifold. The optimizations of these algorithms are based on graphs that provide a discretized approximation to the manifold. Partial alignments of the data sets—obtained from prior knowledge of their manifold structure or from pairwise correspondences of subsets of labeled examples—are completed by integrating supervised signals with unsupervised frameworks for manifold learning. As an illustration of this semisupervised setting, we show how to learn mappings between different data sets of images that are parameterized by the same underlying modes of variability (e.g., pose and viewing angle). The curse of dimensionality in these problems is overcome by exploiting the low dimensional structure of image manifolds.

1 Introduction

Examples of very high-dimensional data such as high-resolution pixel images or large vector-space representations of text documents abound in multimodal data sets. Learning problems involving these data sets are difficult due to the curse of dimensionality and associated large computational demands. However, in many cases, the statistical analysis of these data sets may be tractable due to an underlying low-dimensional manifold structure in the data. Recently, a series of learning algorithms that approximate data manifolds have been developed, such as Isomap [15], locally linear embedding [13], Laplacian eigenmaps [3], Hessian eigenmaps [7], and charting [5]. While these algorithms approach the problem of learning manifolds from an unsupervised perspective; in this paper, we address the problem of establishing a regression between two or more data sets by aligning their underlying manifolds. We show how to align the low-dimensional representations of the data sets given some additional information about the mapping between the data sets. Our algorithm relies upon optimization over a graphical representation of the data, where edges in the graphs are computed to preserve local structure in the data. This optimization yields a common low-dimensional embedding which can then be used to map samples between the disparate data sets.

Two main approaches for alignment of manifolds are presented. In the first approach, additional knowledge about the intrinsic embedding coordinates of some of the samples are used to constrain the alignment. This information about coordinates may be available given knowledge about the data generating process, or when some coordinates are manually assigned to correspond to certain labeled samples. Our algorithm yields a graph embedding where these known coordinates are preserved. Given multiple data sets with such coordinate labels, we show how the underlying data manifolds can be aligned to each other through a common set of coordinates.

In the second approach, we assume that there is no prior knowledge of explicit coordinates, but that we know the pairwise correspondences of some of the samples [11, 16]. These correspondences may be apparent from temporal conjunction, such as simultaneously obtained images and sounds from cameras and microphones. Correspondences may also be obtained from hand-labeled matches among samples in different data sets. We demonstrate how these correspondences allow implicit alignment of the different data manifolds. This is achieved by joining the graph representations of the different data sets and estimating a common low-dimensional embedding over the joined graph.

In Section 2 we first review a graph-based framework for manifold learning algorithms. Section 3 describes

our algorithms for manifold alignment using either prior coordinate knowledge or paired correspondences. Section 4 demonstrates the application of our approach to aligning the pose manifolds of images of different objects. Finally, the utility and future direction of this approach is discussed in Section 5.

2 Unsupervised manifold learning with graphs

Let X and Y be two data sets in high dimensional vector spaces

$$X = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_m\} \subset \mathbb{R}^{D_X}, \quad Y = \{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n\} \subset \mathbb{R}^{D_Y},$$

with $D_X, D_Y \gg 1$. When the data lie close to a low-dimensional manifold embedded in a high dimensional Euclidean space, manifold learning algorithms such as [3] can successfully learn low-dimensional embeddings by constructing a weighted graph that captures local structure in the data. Let $G(\mathcal{V}, \mathcal{E})$ be the graph where the vertices $\mathcal{V}$ correspond to samples in the data and the undirected edges $\mathcal{E}$ denote neighborhood relationships between the vertices. These neighborhood relations can be defined in terms of k-nearest neighbors or an ϵ-ball distance criterion in the Euclidean space of original data. The similarities between points are summarized by a weight matrix W where $W_{ij} \neq 0$ when the ith and jth data points are neighbors ($i \sim j$), otherwise $W_{ij} = 0$. The matrix W is typically symmetric, and has nonnegative weights $W_{ij} = W_{ji} \geq 0$. The generalized graph Laplacian L is then defined as:

$$L_{ij} := \begin{cases} d_i, & \text{if } i = j, \\ -W_{ij}, & \text{if } i \sim j, \\ 0, & \text{otherwise} \end{cases}$$

where $d_i = \sum_{j \sim i} W_{ij}$ is the degree of the ith vertex. If the graph is connected, L will have a single zero eigenvalue associated with the uniform vector $\boldsymbol{e} = [11 \cdots 1]^T$.

A low-dimensional embedding of the data can be computed from the graph Laplacian in the following manner. A real valued function $\boldsymbol{f} : \mathcal{V} \mapsto \mathbb{R}$ on the vertices of the graph is associated with the cost:

$$\boldsymbol{f}^T L \boldsymbol{f} = \frac{1}{2} \sum_{i,j} (f_i - f_j)^2 W_{ij}. \tag{1}$$

An optimal embedding is given by functions $\boldsymbol{f}$ that minimize (1), subject to scale and translation constraints $\boldsymbol{f}^T \boldsymbol{f} = 1$ and $\boldsymbol{f}^T \boldsymbol{e} = 0$. These solutions are then the eigenvectors of L with the smallest non-zero eigenvalues [8]. These solutions may also be interpreted as the kernel principal components of a Gram matrix given by the pseudoinverse of L [10]. This interpretation defines a metric over the graph which is related to the commute times of random walks on the graph [1], and resistance distance in electrical networks [6].

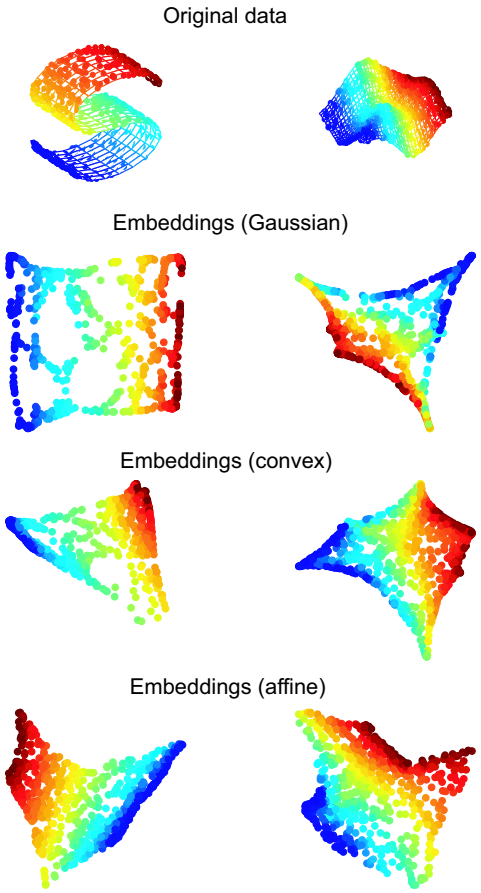


Figure 1: Two-dimensional embeddings of surfaces in $\mathbb{R}^3$. The embeddings are computed from diagonalizing the graph Laplacians. Different edge weightings yield qualitative differences in the embeddings. Only 600 and 800 points were sampled from the two manifolds, making it difficult for the algorithms to find a faithful embedding of the data.

Choice of weights

Within this graph framework, different algorithms may employ different choices for the weights W. For example, W can be defined according to the Gaussian process $W_{ij} = e^{-|\boldsymbol{x}_i - \boldsymbol{x}_j|^2/2\sigma^2}$, and is related to a diffusion process on the graph [3, 12]. The symmetric, nonnegative assumptions on the weights $W_{ij} = W_{ji} \geq 0$ can be relaxed. For a directed graph structure, such as when the neighborhoods are determined by k-nearest neighbors, the matrix W is not symmetric. Nonnegativity constraints may also be lifted. Consider the

least-squares approach to optimize weights W_{ij} :

$$W_{ij} = \arg\min_{W} \ |\boldsymbol{x}_i - \sum_{j\sim i} W_{ij}\boldsymbol{x}_j|^2, \qquad (2)$$

that is, W_{ij} are the coefficients of the neighbors of $\boldsymbol{x}_i$ that best approximates $\boldsymbol{x}_i$, and are in general asymmetric. Locally linear embedding determines weights from minimizing (2) subject to $\sum_j W_{ij} = 1$, yielding possibly negative coefficients that best approximates $\boldsymbol{x}_i$ from an affine combination of its neighbors [14]. This is in contrast to minimizing (2) over a set of convex coefficients that are nonnegative: $W_{ij} \geq 0$. As noted in [14], a possible disadvantage of convex approximation is that a point on the boundary may not be reconstructed from the convex hull of its neighbors. Consequently, the corners of the resultant embedding with convex weights tend to be rounded.

Graph Laplacians with negative weights have been recently studied [9, 10]. Although it is difficult to properly generalize spectral graph theory, we can define a new cost function analogous to (1) for graphs with asymmetric, negative weights as:

$$\boldsymbol{f}^T L^T L \boldsymbol{f} = \sum_i |f_i - \sum_{j\sim i} W_{ij} f_j|^2, \qquad (3)$$

where $L = D - W$. Since $L^T L$ is positive semidefinite and satisfies $L^T L\boldsymbol{e} = 0$, the eigenvectors of $L^T L$ can be used to construct a low-dimensional embedding of the graph that minimizes the cost (3).

Figure 1 shows the unsupervised graph embedding of two artificial data sets using three different weighting schemes: a symmetric Gaussian, asymmetric convex reconstruction, and asymmetric affine reconstruction weights. 600 points were sampled from an S-shaped two-dimensional manifold, and 800 points were sampled from a wavy two-dimensional manifold. The data was intentionally undersampled, and the unsupervised learning algorithms have difficulty in faithfully reconstructing the proper embedding. In the next sections, we will show how semisupervised approaches can greatly improve on these embeddings with the same data.

3 Semisupervised alignment of manifolds

We now consider aligning disparate data manifolds, given some additional information about the data samples. In the following approaches, we consider this additional information to be given for only a partial subset of the data. We denote the samples with this additional "labeled" information by the ordinal index l, and the samples without extra information by the index u. We also use the same notation for the sets X and Y; for example, X_l and Y_l refer to the "labeled" parts of X and Y.

This additional information about the data samples may be of two different types. In the first algorithm, the labels refer to prior information about the intrinsic real-valued coordinates within the manifold for particular data samples. In the second algorithm, the labels indicate pairwise correspondences between samples $\boldsymbol{x}_i \in X$ and $\boldsymbol{y}_j \in Y$. These two types of additional information are quite different, but we show how each can be used to align the different data manifolds.

3.1 Alignment with given coordinates

In this approach, we are given desired coordinates for certain labeled samples. Similar to regression models, we would like to find a map defined on the vertices of the graph $f : V \mapsto \mathbb{R}$ that matches known target values for the labeled vertices. This can be solved by finding $\arg\min_f \ |f_i - s_i|^2 \ (i \in l)$ where $\boldsymbol{s}$ is the vector of target values. With a small number of labeled examples, it is crucial to exploit manifold structure in the data when constructing the class of admissible functions $\boldsymbol{f}$. The symmetric graph Laplacian $L = L^T$ provides this information. A regularized regression cost on a graph is defined as:

$$C(\boldsymbol{f}) = \sum_i \mu|f_i - s_i|^2 + \boldsymbol{f}^T L\boldsymbol{f}. \qquad (4)$$

The first term in (4) is the fitting error, and the second term enforces smoothness along the manifold by $\boldsymbol{f}^T L\boldsymbol{f} \approx \sum_i |\nabla_i \boldsymbol{f}|^2$ [2, 17]. The relative weighting of these terms is given by the coefficient μ. The optimum $\boldsymbol{f}$ is then obtained by the linear solution:

$$\boldsymbol{f} = \begin{pmatrix} \mu I + L_{ll} & L_{lu} \\ L_{ul} & L_{uu} \end{pmatrix}^{-1} \begin{pmatrix} \mu I \\ 0 \end{pmatrix} \boldsymbol{s}, \qquad (5)$$

where L consists of labeled and unlabeled partitions:

$$L = \begin{bmatrix} L_{ll} & L_{lu} \\ L_{ul} & L_{uu} \end{bmatrix}.$$

In the limit $\mu \to \infty$, i.e. there is no uncertainty in the labels $\boldsymbol{s}$, the solution becomes

$$\boldsymbol{f}_u = -(L_{uu})^{-1} L_{ul}\boldsymbol{s} = (L_{uu})^{-1} W_{ul}\ \boldsymbol{s}. \qquad (6)$$

This result is directly related to harmonic functions [18], which are smooth functions on the graph such that f_i is determined by the average of its neighbors:

$$f_i = \frac{\sum_j W_{ij} f_j}{\sum_j W_{ij}}. \qquad (7)$$

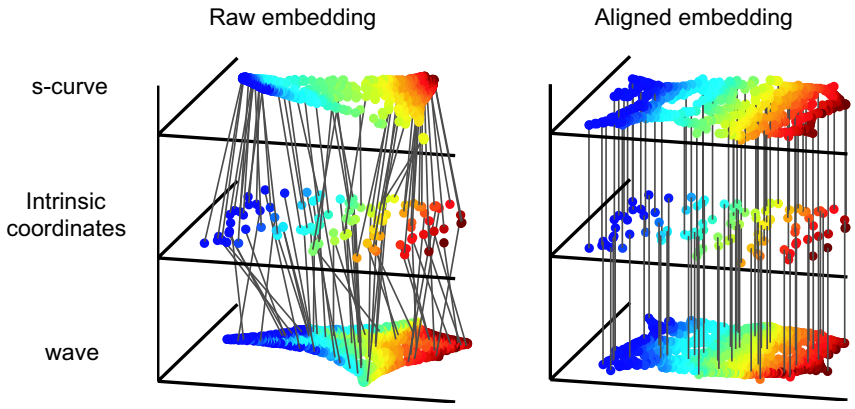


Figure 2: Graph embeddings for the s-curve and wave surface are aligned with given coordinates, and compared to the unaligned embeddings. The lines indicate samples whose known coordinates are used to estimate a common embedding space.

The solution in (6) is a linear superposition of harmonic functions which directly interpolate the labeled data.

Given r-dimensional coordinate vectors $S = [\boldsymbol{s}_1 \cdots \boldsymbol{s}_r]$ as desired embedding coordinates, solutions $\boldsymbol{f}_i$ of (5) or (6) can be used as estimated coordinates of unlabeled data. This "stretches" the embedding of the graph so that the labeled vertices are at the desired coordinates. Figure 3 shows the results of this algorithm applied to an image manifold with two-dimensional pose parameters as coordinates. Simultaneous alignment of two different data sets is performed by simply mapping each of the data sets into a common space with known coordinates. Given two data sets X and Y, where subsets X_l and Y_l are given coordinates $\boldsymbol{s}$ and $\boldsymbol{t}$ respectively, we let $\boldsymbol{f}$ and $\boldsymbol{g}$ denote real-valued functions, and L^x and L^y the graph Laplacians of X and Y respectively. Since there is no explicit coupling between X and Y, we use (6) to get the two solutions:

$$\boldsymbol{f}_u = -(L^x_{uu})^{-1} L^x_{ul}\, \boldsymbol{s}, \quad \text{and} \quad \boldsymbol{g}_u = -(L^y_{uu})^{-1} L^y_{ul}\, \boldsymbol{t}.$$

Figure 2 shows the semisupervised algorithm applied to the synthetic data used in the previous section. Among the 600 and 800 points, 50 labeled points are randomly chosen from each, and the two-dimensional coordinates are provided for $\boldsymbol{s}$ and $\boldsymbol{t}$. The graph weights are chosen by the best convex reconstruction from 6 and 10 neighbors. As can be seen from the figure, the two curves are automatically aligned to each other by sharing a common embedding space. From this common embedding, a point on the s-curve can be mapped to the corresponding point on the wave surface using nearest neighbors, without inferring a direct transformation between the two data spaces.

In [18, 17] the authors assumed symmetric and nonnegative weights. With an asymmetric L, the quadratic term in (4) is no longer valid, and the smoothness term may be replaced with the squared error cost (3). However, there is a difference in the resulting aligned embeddings using a different choice of edge weights on the graph. This is illustrated in the right side of Figure 3 where convex and affine weights are used. With convex weights, the aligned embedding of unlabeled points lies within the convex hull of labeled points. In contrast, the affine weights can extrapolate to points outside the convex hull of the labeled examples. If we consider the matrix of coefficients $M = -(L_{uu})^{-1} L_{ul}$ in (6), it is not difficult to see $\sum_j M_{ij} = 1$ for all i because $\sum_{j\in u} L_{ij} + \sum_{j\in l} L_{ij} = \sum_j L_{ij} = 0$ for all i. Consequently, each row of M are affine coefficients. With an additional constraint $W_{ij} \geq 0$, the M satisfies $M_{ij} \geq 0$ as well, (refer to [4] for proofs) rendering each row of M convex coefficients.

3.2 Alignment by pairwise correspondence

Given multiple data sets containing no additional information about intrinsic coordinates, it is still possible to discover common relationships between the data sets using pairwise correspondences. In particular, two data sets X and Y may have subsets X_l and Y_l which are in pairwise alignment. For example, given sets of images of different persons, we may select pairs with the same pose, facial expression, etc. With this additional information, it is possible to then determine how to match the unlabeled examples using an aligned manifold embedding.

The pairwise correspondences are indicated by the indices $\boldsymbol{x}_i \leftrightarrow \boldsymbol{y}_i$, $(i \in l)$, and $\boldsymbol{f}$ and $\boldsymbol{g}$ denote real-valued functions defined on the respective graphs of X and Y. $\boldsymbol{f}$ and $\boldsymbol{g}$ represent embedding coordinates that are extracted separately for each data set, but they should

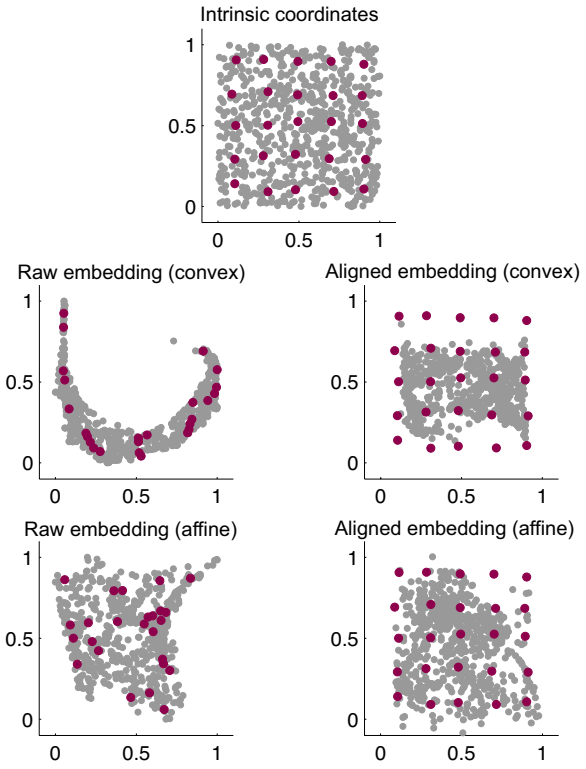


Figure 3: Embedding a data manifold with given coordinates. A set of 698 images of a statue was taken by a camera with varying tilt and pan angles as pose parameters. These pose parameters are provided as labeled coordinates for chosen imges (large dots). This information is used to infer the two-dimensional coordinates corresponding to poses of the unlabeled images. Different conditions for weights in the graph Laplacian result in quite different embeddings.

take similar values for corresponding pairs. Generalizing the single graph embedding algorithm, the dual embedding can be defined by optimizing:

$$C(\boldsymbol{f},\boldsymbol{g}) = \mu \sum_{i\in l} |f_i - g_i|^2 + \boldsymbol{f}^T L^x \boldsymbol{f} + \boldsymbol{g}^T L^y \boldsymbol{g}, \quad (8)$$

where L^x and L^y are the graph Laplacian matrices. The first term penalizes discrepancies between $\boldsymbol{f}$ and $\boldsymbol{g}$ on the corresponding vertices, and the second term imposes smoothness of $\boldsymbol{f}$ and $\boldsymbol{g}$ on the respective graphs.

However, unlike the regression in (4), the optimization in (8) is ill-defined because it is not invariant to simultaneous scaling of $\boldsymbol{f}$ and $\boldsymbol{g}$. We instead should minimize the Rayleigh quotient:

$$\tilde{C}(\boldsymbol{f},\boldsymbol{g}) := \frac{C(\boldsymbol{f},\boldsymbol{g})}{\boldsymbol{f}^T\boldsymbol{f} + \boldsymbol{g}^T\boldsymbol{g}} \quad (9)$$

This quotient can be written in terms of the augmented vector: $\boldsymbol{h} = [\boldsymbol{f}^T \boldsymbol{g}^T]^T$. Minimizing (9) is then equivalent to

$$\min_h \ \tilde{C}(\boldsymbol{h}) := \frac{\boldsymbol{h}^T L^z \boldsymbol{h}}{\boldsymbol{h}^T\boldsymbol{h}}, \ \text{ s.t. } \ \boldsymbol{h}^T\boldsymbol{e} = 0, \quad (10)$$

where L^z is defined as

$$L^z = \begin{bmatrix} L^x + U^x & -U^{xy} \\ -U^{yx} & L^y + U^y \end{bmatrix} \geq 0, \quad (11)$$

and U^x,U^y,U^{xy}, and U^{yx} are matrices having non-zero elements only on the diagonal

$$U_{ij} = \begin{cases} \mu, & i = j \in l \\ 0, & \text{otherwise} \end{cases}$$

The r-dimensional embedding is obtained by the r-nonzero eigenvectors of L^z. A slightly different embedding results from using the normalized cost function (9):

$$\tilde{C}(\boldsymbol{f},\boldsymbol{g}) := \frac{C(\boldsymbol{f},\boldsymbol{g})}{\boldsymbol{f}^T D^x \boldsymbol{f} + \boldsymbol{g}^T D^y \boldsymbol{g}},$$

where D^x and D^y are diagonal matrices corresponding to the vertex degrees $D^x_{ii} = \sum_j W^x_{ij}$ and $D^y_{ii} = \sum_j W^y_{ij}$. This optimization is solved by finding the generalized eigenvectors of L^z and $D^z = \mathrm{diag}(D^x, D^y)$.

In (8) the coefficient μ weights the importance of the correspondence term relative to the smoothness term. In the limit $\mu \to \infty$, the result is equivalent to imposing hard constraints $f_i = g_i$ for $i \in l$. In this limit, the optimization is given by the eigenvalue problem:

$$\tilde{C(\boldsymbol{h})} := \frac{\boldsymbol{h}^T L^z \boldsymbol{h}}{\boldsymbol{h}^T\boldsymbol{h}}, \ \text{ s.t. } \ \boldsymbol{h}^T\boldsymbol{e} = 0, \quad (12)$$

where $\boldsymbol{h}$ and L^z are defined as

$$\boldsymbol{h} = \begin{bmatrix} \boldsymbol{f}_l = \boldsymbol{g}_l \\ \boldsymbol{f}_u \\ \boldsymbol{g}_u \end{bmatrix}, \ L^z = \begin{bmatrix} L^x_{ll} + L^y_{ll} & L^x_{lu} & L^y_{lu} \\ L^x_{ul} & L^x_{uu} & 0 \\ L^y_{ul} & 0 & L^y_{uu} \end{bmatrix}. \quad (13)$$

This formulation results in a smaller eigenvalue problem than (10), and the parameter μ need not be explictly determined.

The two methods in (11) and (13) of constructing a new graph Laplacian L^z can be interpreted as joining two disparate graphs. The former definition of L^z links two graphs by adding edges between paired vertices of the graphs with weights μ, whereas the latter L^z "short-circuits" the paired vertices. In either case, the embedding of the joint graph automatically aligns the two constituent graphs.

Figure 4 shows the alignment of the embeddings of s-curve and wave surfaces via the hard coupling of the graphs. Joining the two graphs not only aligns each other, but also highlights the underlying structure in common, yielding slightly more uniform embeddings than the unsupervised ones.

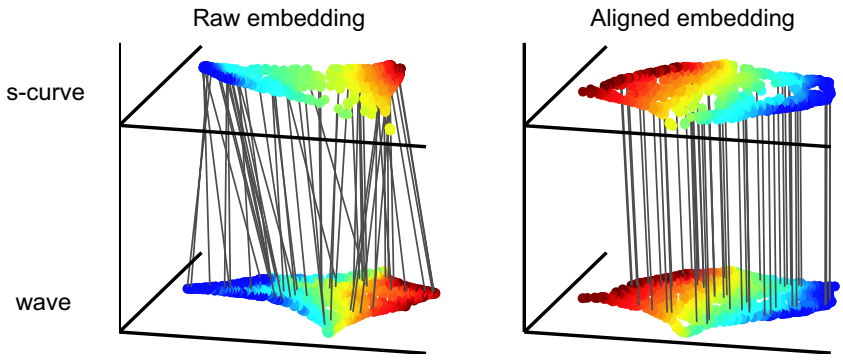


Figure 4: The graph embeddings of the s-curve and wave surface are aligned by pairwise correspondence. 100 pairs of points in one-to-one correspondence are indicated by lines (only 50 shown).

4 Applications

The goal of aligning manifolds was to find an bi-continuous map between the manifolds. A common embedding space is first learned by incorporating additional information about the data samples. We can use this common low-dimensional embedding space to address the following matching problems. What is the most relevant sample $\boldsymbol{y}_i \in Y$ that corresponds to a $\boldsymbol{x}_j \in X$? or the most relevant sample $\boldsymbol{x}_i \in X$ that corresponds to a $\boldsymbol{y}_j \in Y$?

The Euclidean distance of samples in the common embedding space can provide a relevant measure for matching. Let $F = [\boldsymbol{f}_1 \boldsymbol{f}_2 \cdots \boldsymbol{f}_r]$ and $G = [\boldsymbol{g}_1 \boldsymbol{g}_2 \cdots \boldsymbol{g}_r]$ be the r-dimensional representations of aligned manifolds of X and Y. If the coordinates in F and G are aligned from known coordinates, the distance between $\boldsymbol{x}_i \in X$ and $\boldsymbol{y}_j \in Y$ is defined by the usual distance:

$$d(\boldsymbol{x}_i, \boldsymbol{y}_j)^2 := \sum_k |F_{ik} - G_{jk}|^2.$$

If F and G are computed from normalized eigenvectors of a graph Laplacian, the coordinates should be properly scaled. We use the eigenvalues $\lambda_1, \lambda_2, \cdots$, to scale the distance between $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ [10]:

$$d(\boldsymbol{x}_i, \boldsymbol{y}_j)^2 := \sum_k |F_{ik} - G_{jk}|^2 / \lambda_k.$$

Then the best match $\boldsymbol{y}_i \in Y$ to $\boldsymbol{x} \in X$ is given by finding $\arg\min_i d(\boldsymbol{x}, \boldsymbol{y}_i)$.

We demonstrate matching image examples with three sets of high-dimensional images. The three data sets X, Y, and Z consist of 841 images of a person, 698 images of a statue, and 900 images of the earth, available at *http://www.seas.upenn.edu/∼jhham* and *http://isomap.stanford.edu/datasets.html*. Data set X consists of 120×100 images obtained by varying the pose of a person's head with a fixed camera. Data set Y are 64×64 computer generated images of a 3-D model with varying light sources and pan and tilt angles for the observer. Data set Z are 100×100 rendered images of the globe by rotating its azimuthal and elevation angles. For Y and Z we know the intrinsic parameters of the variations: Y varies through -75 to 75 degrees of pan and -10 to 10 degrees of tilt, and -75 to 75 degrees of light source angles. Z contains of -45 to 45 degrees of azimuth and -45 to 45 degrees for elevation changes. We use the pan and tilt angles of Y and Z as the known 2-D coordinates of the embeddings. Affine weights are determined with 12,12, and 6 nearest neighbors to construct the graphs of data X, Y, and Z.

We describe how both known pose coordinates as well as pairwise correspondences are used to align the image manifolds from the three different data sets.

Matching two sets with correspondence and known coordinates

The task is to align X and Y using both the correspondences of $X \leftrightarrow Y$, and the known pose coordinates of Y. First, 25 matching pairs of images in X and Y are manually chosen. The joint graph of X and Y is formed by fusing the corresponding vertices as in (13). Then the joint graph is aligned to the 25 sample coordinates of Y by (6). The best matching images in X and Y that correspond to various pose parameters are found by nearest image samples in the embedding space. Figure 5 shows the result when 16 grid points in the pose parameter embedding are given and the best matching images in X and Y are displayed.

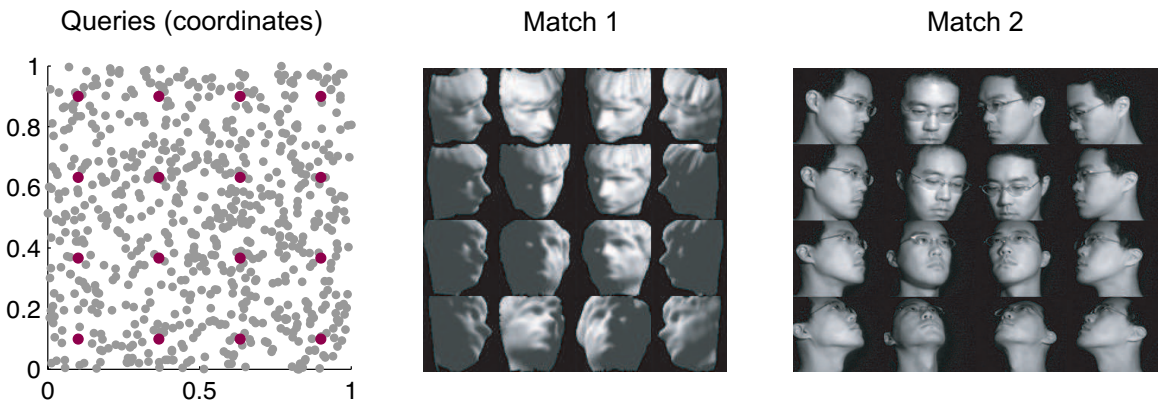


Figure 5: Matching two data sets with correspondence and external coordinates. 25 images of a statue are parameterized by its tilt/pan angles (gray dots on the left). Additionally, 25 corresponding pairs of images of the statue and person are manually matched. Given 16 queries (dark dots on the left) in the embedding space, the best matching images of statue (middle) and person (right) are found by aligning the two data sets and pose parameters simultaneously.

Matching three sets with correspondence

We also demonstrate the simultaneous matching of three data sets. Among the three data sets, we have pairwise correspondence between example images in $X \leftrightarrow Y$ and examples images in $Y \leftrightarrow Z$ separately. 25 pairs of corresponding images between X and Y are used, and an additional 25 pairs of images in Y and Z are chosen manually. The joint graph of X, Y, and Z is formed by the straightforward extension of (12) to handle three sets. A joint graph Laplacian is formed and the final aligned embeddings of the three sets are computed by diagonalizing the graph Laplacian. Given unlabeled sample images from Z as input, the best matching data for Y and X are determined and shown in Figure 6.

5 Discussion

The main computational cost of the graph algorithm lies in finding the spectral decomposition of a large matrix. We employ methods for calculating eigenvectors of large sparse matrices to efficiently speed computation of the embeddings. The graph algorithm is able to quite robustly align the underlying manifold structure in these data sets. Even with the small number of training samples provided, the algorithm is able to estimate a common low-dimensional embedding space which can be used to map samples from one data set to another. Even in situations where the unsupervised manifold learning algorithm suffers from a lack of samples, additional knowledge from the known coordinates and/or pairwise correspondences can be used to discover a faithful embedding. We are currently working on extending these results on additional real-world data sets such as video streams and audio signals.

Finally, we would like to acknowledge support from the U.S. National Science Foundation, Army Reserach Office, and Defense Advanced Research Projects Agency.

References

[1] D. Aldous and J. Fill. Reversible Markov chains and random walks on graphs, 2002. In preparation.

[2] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and regression on large graphs. In *Proceedings of 17th Annual Conference on Learning Theory*, pages 624–638, 2004.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation 15*, pages 1373–1396, 2003.

[4] A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Science*. Academic Press, New York, 1996.

[5] M. Brand. Charting a manifold. In *Advances in Neural Information Processing Systems 15*, pages 961–968, Cambridge, MA, 2003. MIT Press.

[6] A. K. Chandra, P. Raghavan, W. L. Ruzzo, and R. Smolensky. The electrical resistance of a graph captures its commute and cover times. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 574–586. ACM Press, 1989.

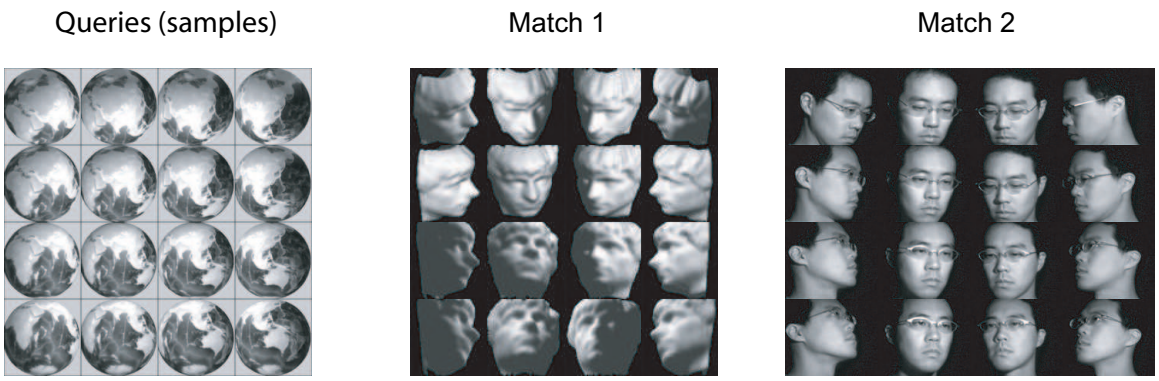


Figure 6: Matching three data sets using correspondence between 25 image pairs of statue and person, and 25 additional image pairs of statue and earth. After the aligned embedding of the joint graph is computed, it is possible to match images across the three data sets. Given the left images of the earth as queries, the right figures show the best matching images of the statue (middle) and person (right).

[7] D. L. Donoho and C. Grimes. Hessian eigenmaps: locally linear embedding techniques for high-dimensional data. In *Proceedings of National Academy of Science, 100 (10)*, pages 5591–5596, 2003.

[8] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its applications to graph theory. *Czechoslovak Math Journal, 25 (100)*, pages 619–633, 1975.

[9] S. Guattery. Graph embeddings, symmetric real matrices, and generalized inverses. Technical Report NASA/CR-1998-208462 ICASE Report No. 98-34, Institute for Computer Applications in Science and Engineering, August 1998.

[10] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. Kernel view of the dimensionality reduction of manifolds. In *Proceedings of International Conference on Machine Learning*, 2004.

[11] J. Ham, D. D. Lee, and L. K. Saul. Learning high-dimensional correspondences from low-dimensional manifolds. In *Workshop on The Continuum from Labeled to Unlabled Data in Machine Learning and Data Mining at Twentieth International Conference on Machine Learning*, pages 34–39, 2003.

[12] I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of International Conference on Machine Learning*, 2002.

[13] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science, 290*, pages 2323–2326, 2000.

[14] L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research, 4*, pages 119–155, 2003.

[15] J. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science, 290*, pages 2319–2323, 2000.

[16] J. J. Verbeek, S. T. Roweis, and N. Vlassis. Nonlinear CCA and PCA by alignment of local models. In *Advances in Neural Information Processing Systems 16*, 2004.

[17] D. Zhou and B. Schölkopf. A regularization framework for learning from graph data. In *Workshop on Statistical Relational Learning at Twenty-first International Conference on Machine Learning*, 2004.

[18] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of International Conference on Machine Learning*, pages 912–919, 2003.

Learning Causally Linked Markov Random Fields

G. E. Hinton, S. Osindero and K. Bao
Department of Computer Science
University of Toronto
Toronto, Canada M5S 3G4

Abstract

We describe a learning procedure for a generative model that contains a hidden Markov Random Field (MRF) which has directed connections to the observable variables. The learning procedure uses a variational approximation for the posterior distribution over the hidden variables. Despite the intractable partition function of the MRF, the weights on the directed connections and the variational approximation itself can be learned by maximizing a lower bound on the log probability of the observed data. The parameters of the MRF are learned by using the mean field version of contrastive divergence [1]. We show that this hybrid model simultaneously learns parts of objects and their inter-relationships from intensity images. We discuss the extension to multiple MRF's linked into in a chain graph by directed connections.

1 Introduction

Generative models are widely used within machine learning. However, in many applications the graphical models involve exclusively causal, or exclusively undirected edges. In this paper we consider models that contain *both* types of edge, and suggest approximate learning methods for such models. The main contribution of this paper is the proposal of combining variational inference with the contrastive divergence algorithm to facilitate learning in systems involving causally linked Markov Random Fields (MRF's). We support our proposal with examples of learning in several domains.

2 Learning Causal Models

One way to make generative models with stochastic hidden variables is to use a directed acyclic graph as shown in Figure 1 (a). The difficulty in learning such "causal" models is that the posterior distribution over the hidden variables is intractable (except in certain special cases such as factor analysis, mixture models, square ICA or graphs that are very sparsely connected). Despite the intractability of the posterior, it is possible to optimize a bound on the log probability of the data by using a simple factorial distribution, $Q(\mathbf{h}|\mathbf{x})$, as an approximation to the true posterior, $P(\mathbf{h}|\mathbf{x})$ over hidden configurations, $\mathbf{h}$, given a data-vector, $\mathbf{x}$. If the hidden variables are binary, a factorial distribution can be represented by assigning a probability, q_j to each hidden variable, j:

$$Q(\mathbf{h}|\mathbf{x}) = \prod_j q_j^{h_j}(1-q_j)^{1-h_j} \tag{1}$$

where h_j is the binary state of hidden unit j in hidden configuration $\mathbf{h}$. Neal and Hinton [2] show that:

$$-\log P(\mathbf{x}) = \mathcal{F}(\mathbf{x}) - \mathrm{KL}(Q(\mathbf{h}|\mathbf{x})||P(\mathbf{h}|\mathbf{x})) \tag{2}$$

where the $\mathcal{F}$ denotes the 'variational free-energy' of the data and is given by

$$\mathcal{F}(\mathbf{x}) = \sum_{\mathbf{h}} Q(\mathbf{h}|\mathbf{x}) \log Q(\mathbf{h}|\mathbf{x}) - \sum_{\mathbf{h}} Q(\mathbf{h}|\mathbf{x}) \log P(\mathbf{h},\mathbf{x}) \tag{3}$$

where $\mathbf{x}$ is a data-vector and $P(\mathbf{h},\mathbf{x})$ is the joint probability of first generating $\mathbf{h}$ from the model, and then generating $\mathbf{x}$ from $\mathbf{h}$.

Since the intractable KL divergence term in equation 2 is non-negative, the variational free-energy, $\mathcal{F}$, gives a tractable upper bound on the negative log probability of the data. Minimizing this bound also has the useful property that it tends to adjust the parameters to make the true posterior distribution as factorial as possible which makes factorial approximate inference work well in the learned model.

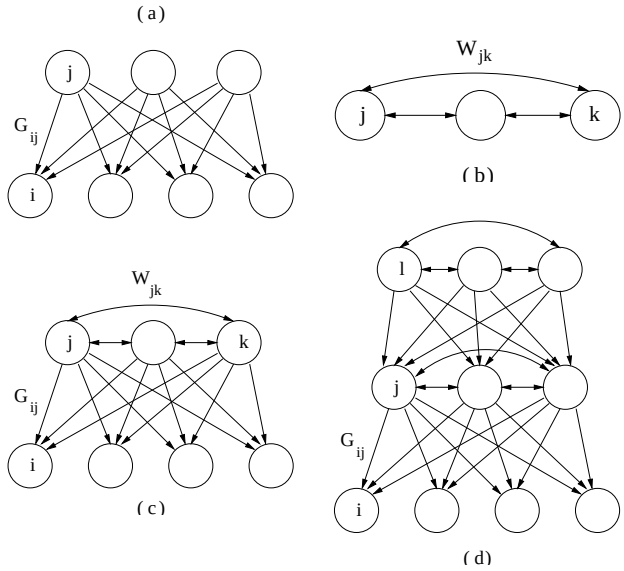


Figure 1: (a) A "causal" generative model. (b) A Markov random field (MRF) with pairwise interactions between the variables. (c) A hybrid model in which the hidden variables of a causal generative model form a Markov random field. (d) A causal hierarchy of MRF's.

For each data-vector in the training set, a locally optimal factorial approximation to the true posterior can be found by following the gradient of the bound w.r.t. Q. Alternatively, the same gradient can be used to train a feedforward "recognition" network to map each training case to a good Q. Once it has been learned, the feedforward network can be viewed as a way of caching the results of iterative settling whilst also acting as a regularizer that encourages similar data-vectors to use similar Q distributions.

3 Learning Markov Random Fields

Hidden latent causes are a good way to model some types of correlation, but they are not good at modeling *constraints* between variables[1]. Consider, for example, a spherical, zero-mean, 20-dimensional Gaussian that has been projected onto the plane in which the sum of the coordinates is 1. To capture this constrained distribution, factor analysis requires 19 hidden factors because it must use a very tight noise model on all 20 variables and then use hidden factors to increase the variance in the 19 allowable directions of variation. Hidden ancestral variables cannot be used to decrease variance[2].

A better way to model constraints is to use an "energy-based" model that associates high energies with data-vectors that violate constraints. The probability of a data-vector is then defined in terms of its energy using the Boltzmann distribution:

$$P(\mathbf{x}) = \frac{e^{-E(\mathbf{x})}}{Z}, \qquad Z = \sum_{\mathbf{u}} e^{-E(\mathbf{u})} \tag{4}$$

where $\mathbf{x}$ is a data-vector, $E(\mathbf{x})$ is its energy, and $\mathbf{u}$ is an index over all possible data-vectors.

The main difficulty in learning energy-based models comes from the normalizing term, Z, (called the partition function) in Eq 4. This is an intractable sum or integral over all possible data-vectors. If a Markov chain is used to sample vectors, $\mathbf{u}$ from the distribution defined by the model, it is possible to get an unbiased estimate of the gradient of the log probability of the data:

$$\frac{\partial \log P(\mathbf{x})}{\partial \theta} = -\frac{\partial E(\mathbf{x})}{\partial \theta} + \sum_{\mathbf{u}} P(\mathbf{u}) \frac{\partial E(\mathbf{u})}{\partial \theta} \tag{5}$$

However, the estimate of the gradient will be very noisy and it is typically hard to know how long to run the Markov chain before it is sampling from the model's distribution. In practice, it is common to assume that if the learning works, the Markov chain must have been close to its equilibrium distribution — a dubious inference.

In some energy-based models, such as a Boltzmann machine with interconnected hidden variables, it is necessary to sum over all possible configurations of the hidden variables to compute the numerator in Eq 4. In other energy-based models, such as "fully visible" Boltzmann machines that just have lateral connections between the visible units it is easy to compute the energy of a data-vector[3] but it is still hard to get the exact derivatives of the partition function. For models of this type, Hinton [3] has shown that learning can still work very well if a Markov chain is started at the data and then run for just a few steps instead of being run all the way to equilibrium.

The use of a brief Markov chain can be combined with the mean field approximation in which the distribution over binary configurations is represented by a factorial distribution Q [1]. For fully visible Boltzmann machines, this leads to a learning algorithm in which the network starts at a data-vector and then updates the q values of all the units in parallel using the rule:

$$q_j^{t+1} = \lambda q_j^t + \frac{1-\lambda}{1+\exp(-b_j - \sum_k q_k^t w_{jk})} \tag{6}$$

[1] In a directed graph, this requires *observed* descendants.

[2] Assuming the factor loadings do not use imaginary components to create negative variance.

[3] Other models that fall within this class include "restricted Boltzmann machines" in which there are no interconnections between hidden units and also models in which the global energy is a function of the activities of multiple layers of deterministic, non-linear hidden units.

where b_j is the bias of unit j, w_{jk} is a symmetric connection between unit j and unit k, and λ is a damping coefficient between 0 and 1 that is used to prevent oscillations. Using the parallel updates in Eq. 6, the learning rule in Eq. 5 becomes:

$$\Delta w_{jk} \propto \sum_{cases} q_j^+ q_k^+ - q_j^- q_k^- \tag{7}$$

where the q^+ values are the the components of a training vector and the q^- values are produced by allowing the mean field net to run for a few iterations of equation 6. The q^+ values would normally be binary, but the learning procedure can still be applied if each training case is a factorial distribution over binary vectors.

4 Causally Linked Markov Random Fields

Both purely causal models and MRF's are used extensively within machine learning, but there are noticeably fewer models in the literature that employ both causal and undirected connections [4]. Causal hierarchies of MRF's (chain-graphs) have some very attractive properties as generative models (see below) but the problem of *learning* them efficiently when there is dense connectivity has not been adequately addressed.

To generate data from such a model[6], we first run the top-level MRF to equilibrium and pick a configuration from the distribution defined by its energy function. This configuration then provides top-down input to the MRF at the next level down via the causal connections. The top-down input modifies the energy function of the second level MRF by changing the effective biases of its units[5]. We then run the second level MRF using its modified energy function and pick a configuration from its distribution. This can be repeated for as many levels as desired, with the bottom level being the "visible" units which may or may not be connected together in an MRF.

This generative model has a major advantage over a purely causal hierarchy: At each level of the hierarchy, learned constraints can be used to "clean-up" the representations generated from the level above. Consider, for example, a generative model in which the top level represents the pose parameters of a face and the next level down represents the pose parameters of each of the two eyes. The height of an eye within the face is somewhat variable, but the two eyes are constrained to have the same height. This creates a problem for a purely causal hierarchy in which the poses of the left and right eye are conditionally independent given the representation at the level above. The height of both eyes must be chosen at the top level and then the height of each eye must be communicated very accurately to the level below. But if an MRF can be used for clean-up at the level below, the height of each eye can be loosely determined by the top-down input, and the MRF can then enforce the constraint on the two heights. So the top-down input to each level can be used to select between (and distort) highly structured and finely balanced alternatives rather than having to specify a pattern in full detail. The causal connections are adept at suggesting which 'parts' to instantiate and roughly where to put them, whilst the undirected connections within the MRF are ideal for enforcing consistency relationships between these parts.

As we shall see, combining multiple MRF's into causal hierarchies also has a major advantage over combining them into one big MRF by using undirected connections: The causal connections between layers act as insulators that prevent the partition functions of the individual MRF's from combining together into one large partition function.

5 A simple version of the model

We begin by presenting the simplest architecture from the framework we have just described: a single, hidden MRF layer with causal connections to a layer of observed variables as illustrated by the network shown in Figure 1 (c).

For concreteness, we will work with a particular simple form for the model's interactions, although more elaborate cases can be treated in essentially the same way. The hidden MRF layer will consist of a Boltzmann machine which has binary nodes with pairwise interaction energies of the form $E(h_i, h_j) = h_i h_j w_{ij}$, and single node energies of the form $E(h_i) = b_i h_i$ where h_k is the binary state of node k and $\{w_{ij}, b_i\}$ are free parameters to be learned. Conditioned upon these hidden variables, the directed connections in our model specify a Gaussian distribution on the observables with $P(\mathbf{x}|\mathbf{h}) = \mathcal{N}(\mathbf{G}\mathbf{h} + \mathbf{m}; \sigma\mathbf{I})$ where σ is a pre-specified noise variance[6].

We use a single-layer sigmoid recognition network to specify the q's of the posterior approximation in equation 1 and the probabilities are given by

$$q_i = \left(1 + e^{-\sum_j R_{ij} x_j + c_i}\right)^{-1} \tag{8}$$

where $\{R_{ij}, c_i\}$ are parameters to be learned [7].

[4] Such models are formally referred to as chain-graphs; see for example [4, 5].

[5] It could also modify pairwise interactions between units in the lower-level MRF.

[6] We fix σ for simplicity, but it could also be learned.

[7] The derivatives that are used to train this recognition

Our formalism leads to the following expression for the variational free energy,

$$\mathcal{F} = \mathcal{F}_{\text{MRF}} + \mathcal{F}_{\text{Gauss}} \quad (9)$$

$$\mathcal{F}_{\text{MRF}} = \sum_i [q_i \log q_i + (1-q_i)\log(1-q_i)] - \frac{1}{2}\mathbf{q}^T\mathbf{W}\mathbf{q} - \mathbf{b}^T\mathbf{q} + \log Z \quad (10)$$

$$\mathcal{F}_{\text{Gauss}} = \frac{1}{2\sigma^2}\left(\mathbf{q}^T\mathbf{G}^T\mathbf{G}\mathbf{q} - 2\mathbf{x}^T\mathbf{G}\mathbf{q}\right) + \mathbf{q}^T\mathbf{K}(1-\mathbf{q}) + c \quad (11)$$

where $K_{ij} = \delta_{ij}(\mathbf{G}^T\mathbf{G})_{ij}$, and c denotes constants that do not affect the derivatives of $\mathcal{F}$ *w.r.t.* the parameters. Minimising $\mathcal{F}$ is equivalent to maximising a lower bound on the data log-likelihood.

A crucial property of this model is that the intractable $\log Z$ term only depends on the biases and lateral connections of the hidden units. It does not enter into the derivatives of either the q^+ values or the weights on the causal connections. So the recognition weights $(\mathbf{R}, \mathbf{c})$ that determine the q values, and also the causal generative parameters $(\mathbf{G}, \mathbf{m})$, can be learned by using the exact gradient of the cost function To learn the hidden biases and the lateral weights $(\mathbf{b}, \mathbf{W})$ between hidden units, we allow the hidden units to run for a few mean field iterations from their initial q values and then use the contrastive divergence learning rule [3], as given in equation 7.

6 A toy example

To illustrate the model we used 50,000 24x24 images of the digit seven that were generated by small rotations, translations and scalings of 1000 normalized 16x16 images from the Cedar CD-Rom. The distortions reduced the long-range correlations introduced by the normalization. We trained a network with 64 fully inter-connected hidden units for 500 sweeps through the training set updating the weights after every 250 examples. There was very little change in the weights after 80 sweeps. We used a momentum of 0.9 with learning rates of 10^{-4} for the causal generative connections and visible biases and for the recognition connections and biases, and 10^{-5} for the lateral connections and hidden generative biases. We also implemented L1 weight-decay corresponding to a Laplacian prior on the lateral connections. This aids interpretability by making most lateral connections small or zero, whilst also allowing large values for a few weights.

Figure 2 (a) and (b) show the generative weights of all 64 hidden units, along with examples of lateral interaction patterns for 4 representative units. The figure caption highlights some salient aspects of the learned lateral connections.

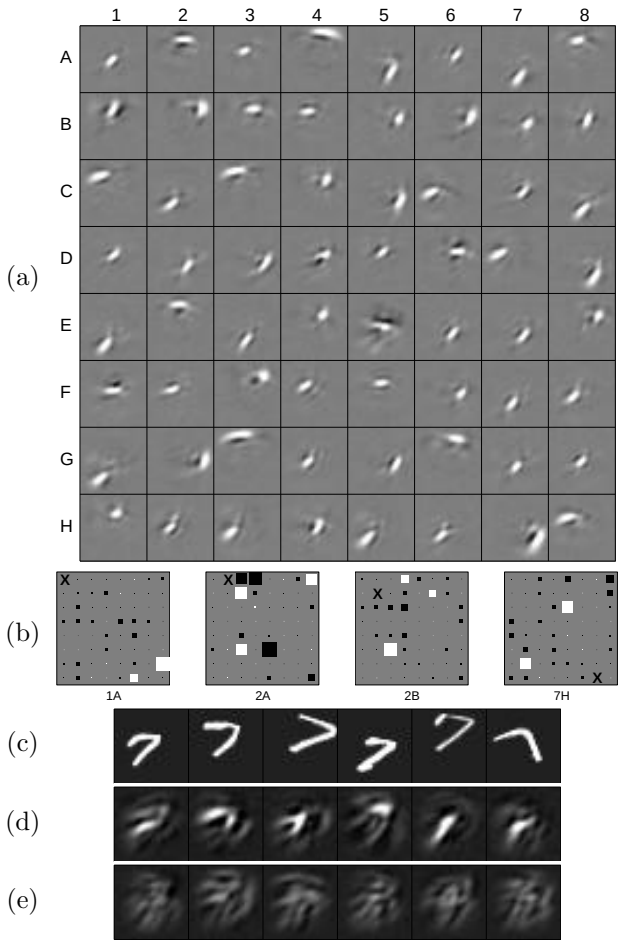


Figure 2: (a) The generative weights of all 64 hidden units in a model of handwritten 7's. (b) The lateral connection patterns for units 1A, 1C, 3F and 7H. The X marks the location of the unit itself. Note the positive interactions between units with collinear generative fields (e.g. 7H and 2G) and also the sizeable negative weights between mutually exclusive alternatives (e.g. 2A and 4A). Unit 2B appears to be a corner detector, and its interactions with 4A and 6B match this intuition. (c) Examples of the training data used. (d) Samples from the distribution learned by the model (obtained using prolonged Gibbs sampling.) (e) Samples from a model with the same generative parameters as in (a,d) but with the lateral connections set to zero, and the biases re-learned to compensate. Notice that there is much less consistency between the strokes in the samples generated from the model without lateral connections.

7 Learning to model natural objects as inter-related parts

It is hard to model real-valued images using binary hidden units so we use binomial units that are equiv-

network could be used to train a far more powerful recognition network that contained hidden layers.

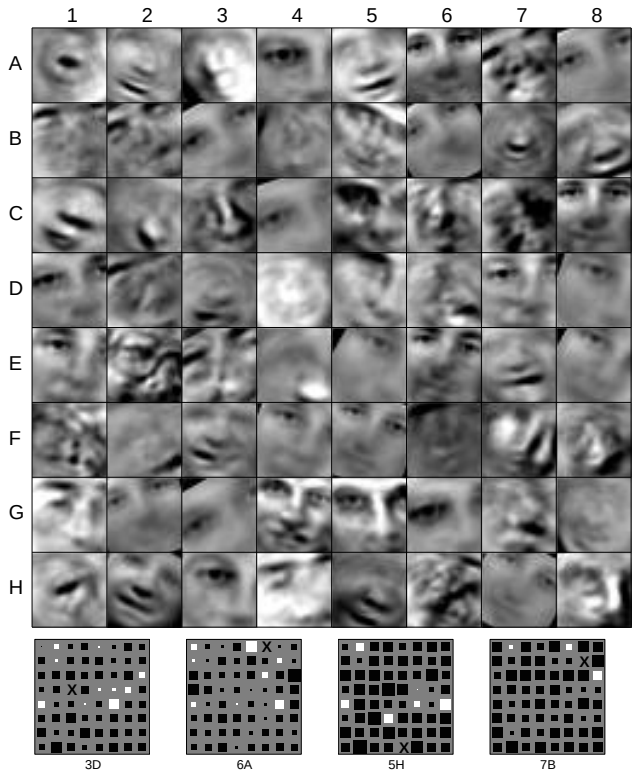


Figure 3: The results of applying the learning algorithm to images of faces. The generative weights of the hidden units are shown at the top and the lateral connections of some of the hidden units are shown beneath. The $8,400$ 31x31 training images were created by rotating ($\pm 30^\circ$), scaling (1.0 to 1.5), cropping and subsampling the 400 face images of 40 different people in the Olivetti face dataset. Each cropped image was then centred (zero pixel mean) and PCA was used to whiten the data and reduce the dimensionality from 961 to 144 by maintaining the normalised projections on the leading 144 eigenvectors.

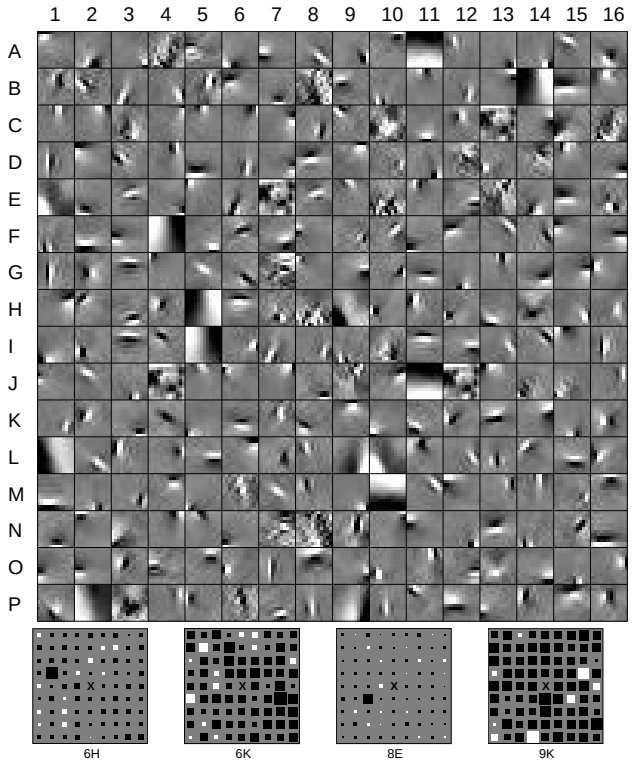


Figure 4: The generative weights of 256 hidden units trained on $150,000$ 12x12 patches of natural images extracted from Hyvarinen's natural image data. The images were whitened and reduced to 100 dimensions using centering and PCA. The lateral interactions were restricted to a 9x9 neighborhood with wraparound. There are strong negative interactions between anti-phase pairs {6H, 3G} & {8E, 6F} and also between highly non-collinear pairs {6H, 5H}, {6K, 5H}, {6K, 5I}, {6K, 9L}. The interactions between approximately collinear pairs with consistent phase are usually positive: {6H, 2D}, {6H, 5G}, {6K, 2L}.

alent to replicating each hidden unit (together with all its weights) $N = 100$ times [7]. We also make an additional modification that is motivated by a desire to produce more neurally plausible representation schemes. The variance contributed by a binomial pool of N binary units each of which has a probability of q of turning on is $Nq(1-q)$. (This appears through the term $\mathbf{q}^T\mathbf{K}(1-\mathbf{q})$ in equation 11.) If we omit the $(1-q)$ term, binomial units cannot use values of q near 1 to achieve low variance and so they learn to use small values of q and behave like Poisson units whose variance is linear in their "firing rate".

Figure 3 shows the weights learned by a network with 64 hidden Poisson units when it was trained on images of faces.

After learning, the hidden activities are sparse with a small subset of the units having activities significantly above their baseline for each image. The ability to learn parts and their relationships simultaneously should make it easier to achieve the goal of finding natural parts of objects in sets of unlabelled images [8], but we have not yet had time to explore this issue in detail. Unlike non-negative matrix factorization [9] our model learns parts without requiring any restrictions on the weights, but it is possible that it would be even better at extracting parts if we restricted the weights on the causal connections to be positive.

Clearly, it would be better to perform some extraction of low level features before attempting to extract inter-related parts of complex objects. Figure 4 shows the results of applying exactly the same algorithm to patches of natural images.

8 Learning with multiple hidden layers

Ideally, a whole hierarchy of features at different levels should be learned cooperatively in order to encourage low-level features to be useful for extracting high-level parts that have consistent inter-relations. Our model is proposed with multiple hidden layers in mind, however we have only just started to investigate this empirically.

We now present the free energy, $\mathcal{F}_2$, for a model with two hidden MRF layers, with the 'top' layer having a directed influence on the layer below (as shown in Figure 1 (d)). If we are able to adequately tackle the extra complexity involved in learning such a model then the generalisation to hierarchies of arbitrary depth involves relatively little extra effort. We will now use h^m and h^t to denote the binary states of hidden units in the middle and top MRF layers respectively. As before, $Q(\mathbf{h}^m|\mathbf{x})$ will denote the a factorial approximation to the posterior probabilities for the MRF units connected to the observables, and we will use $R(\mathbf{h}^t|\mathbf{x})$ to denote the factorial approximation for the MRF units in the top layer.

$$\begin{aligned}\mathcal{F}_2 &= \sum_{\mathbf{h}^t} R(\mathbf{h}^t|\mathbf{x}) \log R(\mathbf{h}^t|\mathbf{x}) \\ &+ \sum_{\mathbf{h}^m} Q(\mathbf{h}^m|\mathbf{x}) \log Q(\mathbf{h}^m|\mathbf{x}) \\ &- \sum_{\mathbf{h}^t} R(\mathbf{h}^t|\mathbf{x}) \log P(\mathbf{h}^t) \\ &- \sum_{\mathbf{h}^t,\mathbf{h}^m} R(\mathbf{h}^t|\mathbf{x}) Q(\mathbf{h}^m|\mathbf{x}) \log P(\mathbf{h}^m|\mathbf{h}^t) \\ &- \sum_{\mathbf{h}^m} Q(\mathbf{h}^m|\mathbf{x}) \log P(\mathbf{x}|\mathbf{h}^m) \qquad (12)\end{aligned}$$

The main difference between this free energy and the one which we have already dealt with is due to the term $\sum_{\mathbf{h}^t,\mathbf{h}^m} R(\mathbf{h}^t|\mathbf{x}) Q(\mathbf{h}^m|\mathbf{x}) \log P(\mathbf{h}^m|\mathbf{h}^t)$. The partition function of the middle layer MRF now depends on the states in the top layer MRF. Consequently we are required to deal with an *expectation* over partition functions as one of the terms within our free energy. Again for concreteness we first present the mathematical form of the free energy for a simple case before discussing an initial approximation for overcoming this difficulty. Our model now involves two Boltzmann machine layers, as illustrated by Figure 1 (d), and conditioning on the states of the top layer provides an additional bias term to the energy function of the layer below. The factorial approximation to the posterior on the middle layer units remains unchanged, and a similar approximation is used for the top level units, specifically $R(\mathbf{h}^t|\mathbf{x}) = \prod_j r_j^{h_j^t}(1-r_j)^{1-h_j^t}$. As before, the observables are given by a Gaussian distribution conditioned on the states of the middle layer units. The free energy is given by,

$$\begin{aligned}\mathcal{F}_2 &= \sum_j [r_j \log r_j + (1-r_j)\log(1-r_j)] \\ &+ \sum_i [q_i \log q_i + (1-q_i)\log(1-q_i)] \\ &- \frac{1}{2}\mathbf{r}^T\mathbf{H}\mathbf{r} - \mathbf{c}^T\mathbf{r} + \log Z_{\text{TOP}} \\ &- \frac{1}{2}\mathbf{q}^T\mathbf{W}\mathbf{q} - (\mathbf{b}+\mathbf{r})^T\mathbf{q} \\ &+ \left\langle \log Z_{\text{MID}}(\mathbf{h}^t) \right\rangle_{\mathbf{h}^t \sim R(\mathbf{h}^t|\mathbf{x})} \\ &+ \mathcal{F}_{\text{Gauss}} \qquad (13)\end{aligned}$$

One strategy is to replace the expectation over partition functions with the partition function evaluated at the expected value of $\mathbf{h}^t$, i.e. at $\mathbf{h}^t = \mathbf{r}$. This can be viewed as a first order Taylor series approximation to $\log Z_{\text{MID}}(\mathbf{h}^t)$ about the mean of $R(\mathbf{h}^t|\mathbf{x})$ (higher order expansions might also be feasible, however the terms are much more complicated.) Such an approximation means that the free energy is no longer a bound on the true log likelihood, however we are at present unaware of any other tractable approximation that would allow us to maintain such a bound.

In this new approximation we use contrastive divergence both to estimate derivatives of the lateral connections and MRF biases, and also to compute a component of the derivative with respect to the top level activities, $\mathbf{r}$. (From the point of view of forming derivatives, the top level units simply act as case dependent biases.)

Preliminary experiments using models with two MRF layers causally linked into a hierarchy indicate that this approximation might be adequate for our gradient based learning. The 'middle' MRF layer typically develops features that are qualitatively similar to those in the single layer case. The 'top' level units tend to sensibly co-activate sets of units in the 'middle' layer, however it is hard to properly characterise the behaviour of units deeper within a densely connected network and their effects are not always apparent simply by studying the generative weights.

To illustrate the increased representational power achieved by adding an additional MRF layer, we present somewhat qualitative results from a simple experiment again using the Cedar digits. Our data consisted of 1100 16×16 images of each class type from 0 to 9 (that is 11000 training examples in total). Figure 5 (a) shows an example of the training data. Using this dataset, we trained two different model architectures: the first had a single hidden Boltzmann machine layer

consisting of 256 fully interconnected units; the second had two hidden Boltzmann machine layers, again with 256 fully interconnected units within each layer, and with directed connections from the top layer providing additional biases to the middle layer. We trained both networks until the changes in parameters were very small (approximately 500 sweeps through the whole data set). Figures 5 (b) and (c) illustrate generative samples from models with one and two hidden MRF layers respectively. From this qualitative comparison it is immediately apparent that the model with a hierarchy of MRF layers has managed to capture more of the statistical structure within the dataset. The generated samples in Figure 5 (b) somewhat resemble single digits, but they are also rather contaminated by additional strokes — as if several digits classes were combined. This contamination is present to a much smaller degree in Figure 5 (c) in which we can see clearer examples of single digits being generated. We speculate that the additional hidden layer is beneficial by providing top down biases to shift the middle layer activities in favour of the strokes for particular digit classes, which might then make the task of ensuring 'stroke consistency' easier for the lateral connections within that layer.

9 Improving the accuracy of approximate inference

There are several reasons why one might wish to use models containing both directed and undirected connections. As discussed in Section 4 they are elegantly able to capture some kinds of statistical structure which would be difficult to capture using connections of just one type. In particular, hierarchies of MRF's have many appealing properties that make them suitable for learning parts-based representations.

Another quite different reason for choosing to combine elements of both kinds of model is to allow approximate inference techniques to work more effectively, and this benefit can be seen in the case of even just a single hidden MRF layer. Many approximate inference techniques assume some simplifying independence relationships, but such relationships generally do not, and cannot, hold in the true posterior. In particular, if the latent variables are assumed to be independent in the prior, an effect known as 'explaining-away' causes those variables to coupled in the posterior [10]. However, somewhat counter-intuitively, it is possible to reduce or eliminate this *posterior* dependence by using a model in which the variables are coupled in the opposite way in the *prior*. The required coupling depends on the parameters, but not on the data.

Our proposed learning method is able to take advantage of this fact, and to work within a space of models for which factorial inference is more accurate than it would be able to be if directed connections alone

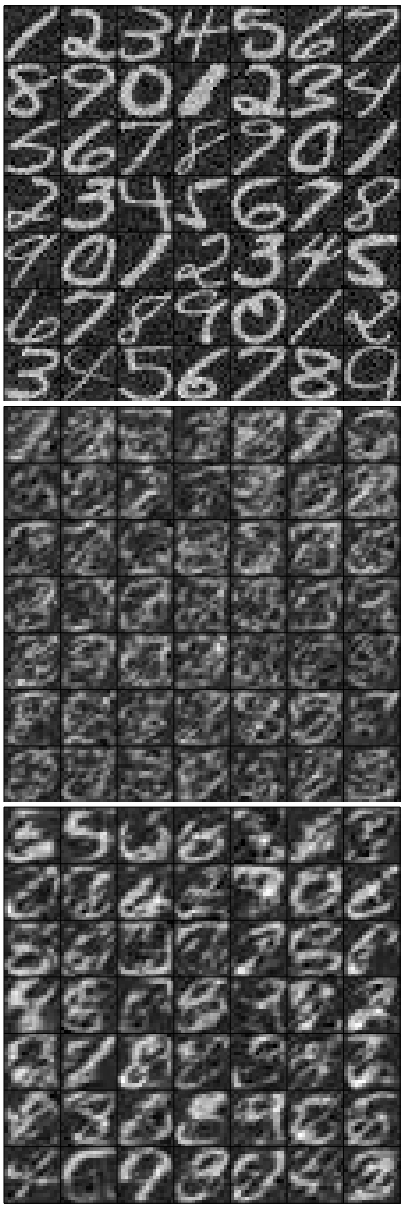


Figure 5: Illustrative results from learning with multiple hidden layers. (a) Examples of training data, corrupted with the same amount of Gaussian noise as assumed during learning. (b) Random selection of examples generated by Gibbs sampling from a model with a single hidden layer. (c) Random selection of examples generated by Gibbs sampling from a model with a hierarchy of two MRF layers. Each MRF layer had 256 fully interconnected hidden units, and there was full directed connectivity from the top MRF layer to the middle MRF layer, as well as full directed connectivity from the middle MRF layer to the observables.

were used. This point is illustrated rather nicely by some of the lateral connections in Figure 4. The lateral interactions tend to cancel out the correlations in the posterior that would be introduced by explaining-away. Consider, for example, two hidden units such as 6H and 3G in Figure 4 that have highly anti-correlated weights on their causal connections. If both these units turn on together the image will be unchanged, so explaining-away would make their activities be strongly positively correlated in the posterior. By learning a strongly negative lateral interaction, the network manages to make them approximately independent in the posterior thus making the variational inference work well.

The idea of using a complicated prior distribution in order to achieve approximate independence in the posterior is a very different approach from Independent Components Analysis (ICA) [11, 12] which assumes independence in the prior and therefore gives rise to awkward posteriors when there are more hidden variables than observables.

10 Summary & Discussion

We have presented a learning procedure for training models that contain both directed and undirected connections; in particular we have focused on large densely connected MRF's that are linked to either observables or other MRF's via directed (causal) connections. Learning in such models is generally intractable, and so the learning task necessitates approximations. Our proposed method combines variational techniques with the contrastive divergence algorithm.

Whilst initial results are promising, there is clearly much more work to be done in developing more sophisticated approximation schemes and in exploring different model architectures for different types of problem. In addition to the approximation methods we have developed in this paper, there are other schemes that may be useful and indeed could be combined with our approach. One could, for instance, consider running our method until convergence and then using this solution as the starting point for a much slower, but potentially more accurate approach that uses Monte Carlo methods. Alternatively, the learned recognition model parameters could be used to initialise further learning using a version of the wake-sleep algorithm [13].

There are many domains in which hybrid models such as the ones we have presented here might be useful, and we hope that our suggested approximation techniques open up avenues for exploration.

Acknowledgements This research was funded by NSERC and CFI. We thank Peter Dayan, Zoubin Ghahramani, Javier Movellan, Sam Roweis, Terry Sejnowski, Yee Whye Teh, Max Welling and Rich Zemel for helpful discussions. GEH holds a Canada Research Chair and is a fellow of CIAR.

References

[1] M. Welling and G. E. Hinton. A new learning algorithm for mean field boltzmann machines. In *Proc. International Conference on Artificial Neural Networks*, pages 351–357. 2002.

[2] R. M. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

[3] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.

[4] S. Lauritzen and N Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics*, 17:31–57, 1989.

[5] W. L. Buntine. Chain graphs for learning. In *Uncertainty in Artificial Intelligence*, pages 46–54, 1995.

[6] S. L. Lauritzen and T. S. Richardson. Chain graphs and their causal interpretations. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, 64(3):321–361, 2002.

[7] Y. W. Teh and G. E. Hinton. Rate-coded restricted boltzmann machines for face recognition. In *Advances in Neural Information Processing Systems 13*. 2001.

[8] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *Proc. 6th European Conference on Computer Vision*, 2000.

[9] D.D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13*. 2001.

[10] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

[11] A. J. Bell and T. J. Sejnowski. An information maximisation approach to blind separation and blind devonvolution. *Neural Cmputation*, 7:1129–1159, 1995.

[12] J.F. Cardoso. Infomax and maximum likelihood for blind source separation. *IEEE Signal Processing Letters*, 4:112–114, 1997.

[13] G.E. Hinton, P. Dayan, B.J. Frey, and R.M. Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268:1158–1160, 1995.

Hilbertian Metrics and Positive Definite Kernels on Probability Measures

Matthias Hein and Olivier Bousquet
Max Planck Institute for Biological Cybernetics, 72076 Tübingen, Germany
{*first.last*}*@tuebingen.mpg.de*

Abstract

We investigate the problem of defining Hilbertian metrics resp. positive definite kernels on probability measures, continuing the work in [5]. This type of kernels has shown very good results in text classification and has a wide range of possible applications. In this paper we extend the two-parameter family of Hilbertian metrics of Topsøe such that it now includes all commonly used Hilbertian metrics on probability measures. This allows us to do model selection among these metrics in an elegant and unified way. Second we investigate further our approach to incorporate similarity information of the probability space into the kernel. The analysis provides a better understanding of these kernels and gives in some cases a more efficient way to compute them. Finally we compare all proposed kernels in two text and two image classification problems.

1 Introduction

Kernel methods have shown in the last years that they are one of the best and generally applicable tools in machine learning. Their great advantage is that positive definite (pd) kernels can be defined on every set. Therefore they can be applied to data of any type. Nevertheless in order to get good results the kernel should be adapted as well as possible to the underlying structure of the input space. This has led in the last years to the definition of kernels on graphs, trees and manifolds. Kernels on probability measures also belong to this category but they are already one level higher since they are not defined on the structures directly but on probability measures on these structures. In recent time they have become quite popular due to the following possible applications:

- Direct application on probability measures e.g. histogram data of text [8] and colors [1].
- Given a statistical model for the data one can first fit the model to the data and then use the kernel to compare two fits, see [8, 7]. Thereby linking parametric and non-parametric models.
- Given a bounded probability space $\mathcal{X}$ one can use the kernel to compare arbitrary sets in that space, e.g by putting the uniform measure on each set.

In this paper we consider Hilbertian metrics and pd kernels on $\mathcal{M}_+^1(\mathcal{X})$[1]. In a first section we will summarize the close connection between Hilbertian metrics and pd kernels so that in general statements for one category can be easily transferred to the other one.
We will consider two types of kernels on probability measures. The first one is general covariant. That means that arbitrary smooth coordinate transformations of the underlying probability space will have no influence on the kernel. Such kernels can be applied if only the probability measures themselves are of interest, but not the space they are defined on. We introduce and extend a two parameter family of covariant pd kernels which encompasses all previously used kernels of this type. Despite the great success of these general covariant kernels in text and image classification, they have some shortcomings. For example for some applications we might have a similarity measure resp. a pd kernel on the probability space which we would like to use for the kernel on probability measures. In the second part we further investigate types of kernels on probability measures which incorporate such a similarity measure, see [5]. This will yield on the one hand a better understanding of these kernels and on the other hand gives in some cases an efficient way of computing these kernels. Finally we apply these kernels on two text (Reuters and WebKB) and two image classification tasks (Corel14 and USPS).

[1] $\mathcal{M}_+^1(\mathcal{X})$ denotes the set of positive measures μ on $\mathcal{X}$ with $\mu(\mathcal{X}) = 1$

2 Hilbertian Metrics versus Positive Definite Kernels

It is a well-known fact that a pd kernel $k(x,y)$ corresponds to an inner product $\langle \phi_x, \phi_y \rangle_{\mathcal{H}}$ in some feature space $\mathcal{H}$. The class of conditionally positive definite (cpd) kernels is less well known. Nevertheless this class is of great interest since Schölkopf showed in [11] that all translation invariant kernel methods can also use the bigger class of cpd kernels. Therefore we give a short summary of this type of kernels and their connection to Hilbertian metrics[2].

Definition 2.1 *A real valued function k on $\mathcal{X} \times \mathcal{X}$ is pd (resp. cpd) if and only if k is symmetric and $\sum_{i,j}^{n} c_i c_j k(x_i, x_j) \geq 0$, for all $n \in \mathbb{N}$, $x_i \in \mathcal{X}, i = 1, ..., n$, and for all $c_i \in \mathbb{R}, i = 1, ..., n$, (resp. for all $c_i \in \mathbb{R}, i = 1, ..., n$, with $\sum_i^n c_i = 0$).*

Note that every pd kernel is also cpd. The close connection between the two classes is shown by the following lemma:

Lemma 2.1 *[2] Let k be a kernel defined as $k(x,y) = \hat{k}(x,y) - \hat{k}(x,x_0) - \hat{k}(x_0,y) + \hat{k}(x_0,x_0)$, where $x_0 \in \mathcal{X}$. Then k is pd if and only if $\hat{k}$ is cpd.*

Similar to pd kernels one can also characterize cpd kernels. Namely one can write all cpd kernels in the form: $k(x,y) = -\frac{1}{2} \|\phi_x - \phi_y\|^2_{\mathcal{H}} + f(x) + f(y)$. The cpd kernels corresponding to Hilbertian (semi)-metrics are characterized by $f(x) = 0$ for all $x \in \mathcal{X}$, whereas if k is pd it follows that $f(x) = \frac{1}{2} k(x,x) \geq 0$. We refer to [2, 3.2] and [11] for further details. We also would like to point out that for SVM's the class of Hilbertian (semi)-metrics is in a sense more important than the class of pd kernels. Namely one can show, see [4], that the solution and optimization problem of the SVM only depends on the Hilbertian (semi)-metric, which is implicitly defined by each pd kernel. Moreover a whole family of pd kernels induces the same semi-metric. In order to avoid confusion we will in general speak of Hilbertian metrics since, using Lemma 2.1, one can always define a corresponding pd kernel. Nevertheless for the convenience of the reader we will often explicitly state the corresponding pd kernels.

[2] A (semi)-metric $d(x,y)$ (A semi-metric $d(x,y)$ fulfills the conditions of a metric except that $d(x,y) = 0$ does not imply $x = y$.) is called Hilbertian if one can embed the (semi)-metric space $(\mathcal{X}, d)$ isometrically into a Hilbert space. A (semi)-metric d is Hilbertian if and only if $-d^2(x,y)$ is cpd. That is a classical result of Schoenberg.

3 γ-homogeneous Hilbertian Metrics and Positive Definite Kernels on $\mathbb{R}_+$[3]

The class of Hilbertian metrics on probability measures we consider in this paper are based on a pointwise comparison of the densities $p(x)$ with a Hilbertian metric on $\mathbb{R}_+$. Therefore Hilbertian metrics on $\mathbb{R}_+$ are the basic ingredient of our approach. In principle we could use any Hilbertian metric on $\mathbb{R}_+$, but as we will explain later we require the metric on probability measures to have a certain property. This in turn requires that the Hilbertian metric on $\mathbb{R}_+$ is γ-homogeneous[4]. The class of γ-homogeneous Hilbertian metrics on $\mathbb{R}_+$ was recently characterized by Fuglede:

Theorem 3.1 (Fuglede [3]) *A symmetric function $d : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}_+$ with $d(x,y) = 0 \iff x = y$ is a γ-homogeneous, continuous Hilbertian metric d on $\mathbb{R}_+$ if and only if there exists a (necessarily unique) non-zero bounded measure $\rho \geq 0$ on $\mathbb{R}_+$ such that d^2 can be written as*

$$d^2(x,y) = \int_{\mathbb{R}_+} \left| x^{(\gamma + i\lambda)} - y^{(\gamma + i\lambda)} \right|^2 d\rho(\lambda) \qquad (1)$$

Using Lemma 2.1 we define the corresponding class of pd kernels on $\mathbb{R}_+$ by choosing $x_0 = 0$. We will see later that this corresponds to choosing the zero-measure as origin of the RKHS.

Corollary 3.1 *A symmetric function $k : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}_+$ with $k(x,x) = 0 \iff x = 0$ is a 2γ-homogeneous continuous pd kernel k on $\mathbb{R}_+$ if and only if there exists a (necessarily unique) non-zero bounded symmetric measure $\kappa \geq 0$ on $\mathbb{R}$ such that k is given as*

$$k(x,y) = \int_{\mathbb{R}} x^{(\gamma + i\lambda)} y^{(\gamma - i\lambda)} \, d\kappa(\lambda) \qquad (2)$$

Proof: If k has the form given in (2), then it is obviously 2γ-homogeneous and since $k(x,x) = x^{2\gamma} \kappa(\mathbb{R})$ we have $k(x,x) = 0 \iff x = 0$. The other direction follows by first noting that $k(0,0) = \langle \phi_0, \phi_0 \rangle = 0$ and then by applying theorem 3.1, where κ is the symmetrized version of ρ around the origin, together with lemma 2.1 and $k(x,y) = \langle \phi_x, \phi_y \rangle = \frac{1}{2} \left(-d^2(x,y) + d^2(x,0) + d^2(y,0) \right)$. □

At first glance Theorem 3.1, though mathematically beautiful, seems not to be very helpful from the viewpoint of applications. But as we will show in the section on structural pd kernels on $\mathcal{M}_+^1(\mathcal{X})$ this result allows us to compute this class of kernels very efficiently.

[3] $\mathbb{R}_+$ is the positive part of the real line with 0 included

[4] A symmetric function k is γ-homogeneous if $k(c\,x, c\,y) = c^\gamma k(x,y)$ for all $c \in \mathbb{R}_+$

Recently Topsøe and Fuglede proposed an interesting two-parameter family of Hilbertian metrics on $\mathbb{R}_+$ [13, 3]. We extend now the parameter range of this family. This allows us in the next section to recover all previously used Hilbertian metrics on $\mathcal{M}_+^1(\mathcal{X})$ from this family.

Theorem 3.2 *The function $d : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}$ defined as:*

$$d^2_{\alpha|\beta}(x,y) = \frac{2^{\frac{1}{\beta}}\left(x^\alpha + y^\alpha\right)^{\frac{1}{\alpha}} - 2^{\frac{1}{\alpha}}\left(x^\beta + y^\beta\right)^{\frac{1}{\beta}}}{2^{\frac{1}{\alpha}} - 2^{\frac{1}{\beta}}} \qquad (3)$$

is a $1/2$-homogeneous Hilbertian metric on $\mathbb{R}_+$, if $\alpha \in [1,\infty]$, $\beta \in [\frac{1}{2}, \alpha]$ or $\beta \in [-\infty, -1]$. Moreover the pointwise limit for $\alpha \to \beta$ is given as:

$$\lim_{\alpha\to\beta} d^2_{\alpha|\beta}(x,y) = \frac{\beta^2 2^{1/\beta}}{\log(2)} \frac{\partial}{\partial\beta}\left(\frac{x^\beta + y^\beta}{2}\right)^{(1/\beta)} =$$

$$\frac{\left(x^\beta + y^\beta\right)^{\frac{1}{\beta}}}{\log(2)}\left[\frac{x^\beta}{x^\beta + y^\beta}\log\left(\frac{2x^\beta}{x^\beta + y^\beta}\right) + \frac{y^\beta}{x^\beta + y^\beta}\log\left(\frac{2y^\beta}{x^\beta + y^\beta}\right)\right]$$

Note that $d^2_{\alpha|\beta} = d^2_{\beta|\alpha}$. We need the following lemmas in the proof:

Lemma 3.1 *[2, 2.10] If $k : \mathcal{X}\times\mathcal{X}$ is cpd and $k(x,x) \leq 0$, $\forall x \in \mathcal{X}$ then $-(-k)^\gamma$ is also cpd for $0 < \gamma \leq 1$.*

Lemma 3.2 *If $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is cpd and $k(x,y) < 0$, $\forall\, x, y \in \mathcal{X}$, then $-1/k$ is pd.*

Proof: It follows from Theorem 2.3 in [2] that if $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_-$ is cpd, then $1/(t-k)$ is pd for all $t > 0$. The pointwise limit of a sequence of cpd resp. pd kernels is cpd resp. pd if the limit exists, see e.g. [10]. Therefore $\lim_{t\to 0} 1/(t-k) = -1/k$ is positive definite if k is strictly negative. □

We can now prove Theorem 3.2:

Proof: The proof for the symmetry, the limit $\alpha \to \beta$ and the parameter range $1 \leq \alpha \leq \infty$, $1/2 \leq \beta \leq \alpha$ can be found in [3]. We prove that $-d^2_{\alpha|\beta}$ is cpd for $1 \leq \alpha \leq \infty$, $-\infty \leq \beta \leq -1$. First note that $k(x,y) = -(f(x)+f(y))$ is cpd on $\mathbb{R}_+$, for any function $f : \mathbb{R}_+ \to \mathbb{R}_+$ and satisfies $k(x,y) \leq 0$, $\forall\, x, y \in \mathcal{X}$. Therefore by Lemma 3.1, $-(x^\alpha + y^\alpha)^{1/\alpha}$ is cpd for $1 \leq \alpha < \infty$. The pointwise limit $\lim_{\alpha\to\infty} -(x^\alpha + y^\alpha)^{1/\alpha} = -\max\{x, y\}$ exists, therefore we can include the limit $\alpha = \infty$. Next we consider $k(x,y) = -(x+y)^{1/\beta}$ for $1 \leq \beta \leq \infty$ which is cpd as we have shown and strictly negative if we restrict k to $\{x \in \mathbb{R} \,|\, x > 0\} \times \{x \in \mathbb{R} \,|\, x > 0\}$. Then all conditions for lemma 3.2 are fulfilled, so that $k(x,y) = (x+y)^{-1/\beta}$ is pd. But then also $k(x,y) = (x^{-\beta}+y^{-\beta})^{-1/\beta}$ is pd. Moreover k can be continuously extended to 0 by $k(x,y) = 0$ for $x = 0$ or $y = 0$. Multiplying the first part with $(2^{(1/\alpha - 1/\beta)} - 1)^{-1}$ and the second one with $(1 - 2^{(1/\beta - 1/\alpha)})^{-1}$ and adding them gives the result. □

4 Covariant Hilbertian Metrics on $\mathcal{M}_+^1(\mathcal{X})$

In this section we define Hilbertian metrics on $\mathcal{M}_+^1(\mathcal{X})$ by comparing the densities pointwise with a Hilbertian metric on $\mathbb{R}_+$ and integrating these distances over $\mathcal{X}$. Since densities can only be defined with respect to a dominating measure[5] our definition will at first depend on the choice of the dominating measure. This dependence would restrict the applicability of our approach. For example if we had $\mathcal{X} = \mathbb{R}^n$ and chose μ to be the Lebesgue measure, then we could not deal with Dirac measures δ_x since they are not dominated by the Lebesgue measure.

Therefore we construct the Hilbertian metric such that it is independent of the dominating measure. This justifies the term 'covariant' since independence from the dominating measure also yields invariance from arbitrary one-to-one coordinate transformations. In turn this also implies that all structural properties of the probability space will be ignored so that the metric on $\mathcal{M}_+^1(\mathcal{X})$ only depends on the probability measures. As an example take the color histograms of images. Covariance here means that the choice of the underlying color space say RGB, HSV or CIE Lab does not influence our metric, since these color spaces are all related by one-to-one transformations. Note however that in practice the results will usually slightly differ due to different discretizations of the color space.

In order to simplify the notation we define $p(x)$ to be the Radon-Nikodym derivative $(dP/d\mu)(x)$ [6] of P with respect to the dominating measure μ.

Proposition 4.1 *Let P and Q be two probability measures on $\mathcal{X}$, μ an arbitrary dominating measure[7] of P and Q and $d_{\mathbb{R}_+}$ a $1/2$-homogeneous Hilbertian metric on $\mathbb{R}_+$. Then $D_{\mathcal{M}_+^1(\mathcal{X})}$ defined as*

$$D^2_{\mathcal{M}_+^1(\mathcal{X})}(P,Q) := \int_{\mathcal{X}} d^2_{\mathbb{R}_+}(p(x), q(x))d\mu(x)\,, \qquad (4)$$

is a Hilbertian metric on $\mathcal{M}_+^1(\mathcal{X})$. $D_{\mathcal{M}_+^1(\mathcal{X})}$ is independent of the dominating measure μ.

For a proof, see [5]. Note that if we use an arbitrary metric on $\mathbb{R}_+$ in the above proposition, we also get a Hilbertian metric. But this metric would only be defined on the set of measures dominated by a certain measure μ and not on $\mathcal{M}_+^1(\mathcal{X})$. Moreover it would also depend on the choice of the dominating measure μ.

[5] A measure μ dominates a measure ν if $\mu(E) > 0$ whenever $\nu(E) > 0$ for all measurable sets $E \subset \mathcal{X}$. In $\mathbb{R}^n$ the dominating measure μ is usually the Lebesgue measure.

[6] In case of $\mathcal{X} = \mathbb{R}^n$ and when μ is the Lebesgue measure we can think of $p(x)$ as the normal density function.

[7] Such a dominating measure always exists take e.g. $M = (P+Q)/2$

We can now apply this principle of building covariant Hilbertian metrics on $\mathcal{M}_+^1(\mathcal{X})$ and use the family of 1/2-homogeneous Hilbertian metrics $d^2_{\alpha|\beta}$ on $\mathbb{R}_+$ from the previous section. This yields as special cases the following well-known measures on $\mathcal{M}_+^1(\mathcal{X})$.

$$\begin{aligned}
D^2_{1|-1}(P,Q) &= \int_{\mathcal{X}} \frac{(p(x)-q(x))^2}{p(x)+q(x)} d\mu(x),\\
D^2_{\frac{1}{2}|1}(P,Q) &= \int_{\mathcal{X}} (\sqrt{p(x)}-\sqrt{q(x)})^2 d\mu(x),\\
D^2_{1|1}(P,Q) &= \frac{1}{\log(2)} \int_{\mathcal{X}} p(x) \log\left[\frac{2p(x)}{p(x)+q(x)}\right]\\
&\quad + q(x)\log\left[\frac{2q(x)}{p(x)+q(x)}\right] d\mu(x),\\
D^2_{\infty|1}(P,Q) &= \int_{\mathcal{X}} |p(x)-q(x)| d\mu(x). \qquad (5)
\end{aligned}$$

$D^2_{1|-1}$ is the symmetric χ^2-measure, $D_{\frac{1}{2}|1}$ the Hellinger distance, $D^2_{1|1}$ the Jensen-Shannon divergence and $D^2_{\infty|1}$ the total variation. The symmetric χ^2-metric was for some time wrongly assumed to be pd and is new in this family due to our extension of $d^2_{\alpha|\beta}$ to negative values of β. The Hellinger metric is well known in the statistics community and was for example used in [7]. The total variation was implicitly used in SVM's through a pd counterpart which we will give below. Finally the Jensen-Shannon divergence is very interesting since it is a symmetric and smoothed variant of the Kullback-Leibler divergence. Instead of the work in [9] where they have a heuristic approach to get from the Kullback-Leibler divergence to a pd matrix, the Jensen-Shannon divergence is a theoretically sound alternative. Note that the family $d^2_{\alpha|\beta}$ is designed in such a way that the maximal distance of $D^2_{\alpha|\beta}$ is $2, \forall\, \alpha, \beta$. For completeness we also give the corresponding pd kernels on $\mathcal{M}_+^1(\mathcal{X})$, where we take in Lemma 2.1 the zero measure as x_0 in $\mathcal{M}_+^1(\mathcal{X})$. This choice seems strange at first since we are dealing with probability measures. But in fact the whole framework presented in this paper can easily be extended to all finite, positive measures on $\mathcal{X}$. For this set the zero measure is a natural choice of the origin.

$$\begin{aligned}
K_{1|-1}(P,Q) &= \int_{\mathcal{X}} \frac{p(x)q(x)}{p(x)+q(x)} d\mu(x),\\
K_{\frac{1}{2}|1}(P,Q) &= \int_{\mathcal{X}} \sqrt{p(x)q(x)} d\mu(x),\\
K_{1|1}(P,Q) &= \frac{-1}{\log(2)} \int_{\mathcal{X}} p(x)\log\left(\frac{p(x)}{p(x)+q(x)}\right)\\
&\quad + q(x)\log\left(\frac{q(x)}{p(x)+q(x)}\right) d\mu(x),\\
K_{\infty|1}(P,Q) &= \int_{\mathcal{X}} \min\{p(x),q(x)\} d\mu(x). \qquad (6)
\end{aligned}$$

The astonishing fact is that we find the four (partially) previously used Hilbertian metrics resp. pd kernels on $\mathcal{M}_+^1(\mathcal{X})$ as special cases of a two-parameter family of Hilbertian metrics resp. pd kernels on $\mathcal{M}_+^1(\mathcal{X})$. Due to the symmetry of $d^2_{\alpha|\beta}$ (which implies symmetry of $D^2_{\alpha|\beta}$) we can even see all of them as special cases of the family restricted to $\alpha = 1$. This on the one hand shows the close relation of these metrics among each other and on the other hand gives us the opportunity to do model selection in this one-parameter family of Hilbertian metrics. Yielding an elegant way to handle both the known similarity measures and intermediate ones in the same framework.

5 Structural Positive Definite Kernels

The covariant Hilbertian metrics proposed in the last section have the advantage that they only compare the probability measures, thereby ignoring all structural properties of the probability space. On the other hand there exist cases where we have a reasonable similarity measure on the space $\mathcal{X}$, which we would like to be incorporated into the metric. We will consider in this section two ways of doing this.

5.1 Structural Kernel I

To incorporate structural information about the probability space $\mathcal{X}$ is helpful when we compare probability measures with disjoint support. For the covariant metrics disjoint measures have always maximal distance, irrespectively how "close" or "far" their support is. Obviously if our training set consists only of disjoint measures learning is not possible with covariant metrics. We have proposed in [5] a positive definite kernel which incorporates a given similarity measure, namely a pd kernel, on the probability space. The only disadvantage is that this kernel is not invariant with respect to the dominating measure. That means we can only define it for the subset $\mathcal{M}_+^1(\mathcal{X},\mu) \subset \mathcal{M}_+^1(\mathcal{X})$ of measures dominated by μ. On the other hand in some cases one has anyway a preferred measure like e.g. for Riemannian manifolds where there exists a natural volume measure. Such a preferred measure is then a natural choice for the dominating measure, so that theoretically it does not seem to be a major restriction. For our experiments it does not make any difference since we anyway use only probabilities over finite, discrete spaces, so that the uniform measure dominates all other measures and therefore $\mathcal{M}_+^1(\mathcal{X},\mu) \equiv \mathcal{M}_+^1(\mathcal{X})$.

Theorem 5.1 (Structural Kernel I) *Let k be a bounded PD kernel on $\mathcal{X}$ and $\hat{k}$ a bounded PD kernel on $\mathbb{R}_+$. Then*

$$K_I(P,Q) = \int_{\mathcal{X}}\int_{\mathcal{X}} k(x,y)\, \hat{k}(p(x),q(y))\, d\mu(x)\, d\mu(y) \qquad (7)$$

is a pd kernel on $\mathcal{M}_+^1(\mathcal{X},\mu)\times\mathcal{M}_+^1(\mathcal{X},\mu)$.

We refer to [5] for the proof. Note that this kernel can easily be extended to all bounded, signed measures as it is in general true for all metrics resp. kernels in this paper. This structural kernel generalizes previous work done by Suquet, see [12], where the special case with $\hat{k}(p(x),q(y)) = p(x)q(y)$ has been considered. The advantage of this choice for $\hat{k}$ is that $K_I(P,Q)$ becomes independent of the dominating measure. In fact it is easy to see that among the family of structural kernels $K_I(P,Q)$ of the form (7) this choice of $\hat{k}$ yields the only structural kernel $K(P,Q)$ which is independent of the dominating measure. Indeed for independence bilinearity of $\hat{k}$ is required, which yields $\hat{k}(x,y) = xy\,\hat{k}(1,1)$.
The structural kernel has the disadvantage that the computational cost increases dramatically compared to the covariant one, since one has to integrate twice over $\mathcal{X}$. An implementation seems therefore only to be possible for either very localized probability measures or a sharply concentrated similarity kernel $\hat{k}$ e.g. a compactly supported radial basis function on $\mathbb{R}^n$.
The following equivalent representation of this kernel will provide a better understanding and at the same time will show a way to reduce the computational cost considerably.

Proposition 5.1 *The kernel $K_I(P,Q)$ can be equivalently written as the inner product in $L_2(T\times S,\omega\otimes\kappa)$:*

$$K_I(P,Q) = \int_T\int_S \phi_P(t,\lambda)\overline{\phi_Q(t,\lambda)}\,d\kappa(\lambda)\,d\omega(t)$$

for some sets T,S with the feature map:

$$\phi : \mathcal{M}_+^1(\mathcal{X},\mu) \to L_2(T\times S,\omega\otimes\kappa),$$
$$P \to \phi_P(t,\lambda) = \int_{\mathcal{X}} \Gamma(x,t)\Psi(p(x),s)d\mu(x).$$

where

$$k(x,y) = \int_T \Gamma(x,t)\overline{\Gamma(y,t)}d\omega(t),$$
$$\hat{k}(p(x),q(x)) = \int_S \Psi(p(x),s)\overline{\Psi(p(y),s)}d\kappa(s).$$

Proof: First note that one can write every pd kernel in the form : $k(x,y) = \langle\Gamma(x,\cdot),\Gamma(y,\cdot)\rangle_{L_2(T,\omega)} = \int_T \Gamma(x,t)\overline{\Gamma(y,t)}d\omega(t)$, where $\Gamma(x,\cdot)\in L_2(T,\mu)$ for each $x\in\mathcal{X}$. In general the space T is very big, since one can show that such a representation always exists in $L_2(\mathbb{R}^{\mathcal{X}},\mu)$, see e.g. [6]. For the product of two positive definite kernels we have such a representation on the set $T\times S$. Since for any finite measure space $(\mathcal{Y},\mu)$ one has $L_2(\mathcal{Y},\mu)\subset L_1(\mathcal{Y},\mu)$ we can apply Fubini's theorem and interchange the integration order. The definition of the feature map $\Phi_P(t,\lambda)$ then follows easily. □

This representation has several advantages. First the functions $\Gamma(x,t)$ give us a better idea what properties of the measure P are used in the structural kernel. Second in the case where $S\times T$ is of the same or smaller size than $\mathcal{X}$ we can decrease the computation cost, since we now have to do only an integration over $T\times S$ instead of an integration over $\mathcal{X}\times\mathcal{X}$. Finally this representation is a good starting point if one wants to approximate the structural kernel. Since any discretization of T,S, or $\mathcal{X}$ or integration over smaller subsets, will nevertheless give a pd kernel in the end.
We illustrate this result with a simple example. We take $\mathcal{X}=\mathbb{R}^n$ and $k(x,y)=k(x-y)$ to be a translation invariant kernel, furthermore we take $\hat{k}(p(x),q(y)) = p(x)q(y)$. The characterization of translation invariant kernels on $\mathbb{R}^n$ is a classical result due to Bochner:

Theorem 5.2 *A continuous function $k(x,y)=k(x-y)$ is pd on $\mathbb{R}^n$ if and only if $k(x-y) = \int_{\mathbb{R}^n} e^{i\langle t,x-y\rangle}d\omega(t)$, where ω is a finite non-negative measure on $\mathbb{R}^n$.*

Obviously we have in this case $T=\mathbb{R}^n$. Then the above proposition tells us that we are effectively computing the following feature vector for each P, $\phi_P(t) = \int_{\mathbb{R}^n} e^{i\langle x,t\rangle}p(x)d\mu(x) = E_P e^{i\langle x,t\rangle}$. Finally the structural kernel can in this case be equivalently written as $K_I(P,Q) = \int_{\mathbb{R}^n} E_P e^{i\langle x,t\rangle}E_Q e^{i\langle x,t\rangle}d\omega(t)$. That means the kernel is in this case nothing else than the inner product between the characteristic functions of the measures in $L_2(\mathbb{R}^n,\omega)$[8]. Moreover the computational cost has decreased dramatically, since we only have to integrate over $T=\mathbb{R}^n$ instead of $\mathbb{R}^n\times\mathbb{R}^n$. Therefore in this case the kernel computation has the same computational complexity as in the case of the covariant kernels. The calculation of the features, here the characteristic functions, can be done as a preprocessing step for each measure.

5.2 Structural Kernel II

The second structural kernel we propose has almost the opposite properties compared to the first one. It is invariant with respect to the dominating measure and therefore defined on the set of all probability measures $\mathcal{M}_+^1(\mathcal{X})$. On the other hand it can also incorporate a similarity function on $\mathcal{X}$, but the distance of disjoint measures will not correspond to their 'closeness' in $\mathcal{X}$.

Theorem 5.3 (Structural Kernel II) *Let $s:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$ be a non-negative function, $\hat{k}$ a one-homogeneous pd kernel on $\mathbb{R}_+$ and μ a dominating*

[8] Note that ω is not the Lebesgue measure.

measure of P and Q. Then

$$K_{II}(P,Q) = \int_{\mathcal{X}^2} s(x,y)\hat{k}(p(x),q(x))\hat{k}(p(y),q(y))d\mu(x)\,d\mu(y), \tag{8}$$

is a pd kernel on $\mathcal{M}_+^1(\mathcal{X})$. K_{II} is independent of the dominating measure. Moreover $K_{II}(P,Q) \geq 0,\ \forall\, P,Q \in \mathcal{M}_+^1(\mathcal{X})$ if $s(x,y)$ is a bounded positive definite kernel.

Proof: We first prove that K_{II} is positive definite on $\mathcal{M}_+^1(\mathcal{X})$. Note that $\sum_{i,j=1}^n c_i c_j K_{II}(P_i,P_j) = \int_{\mathcal{X}^2} s(x,y) \sum_{i,j=1}^n c_i c_j \hat{k}(p_i(x),p_j(x))\hat{k}(p_i(y),p_j(y))d\mu(x)d\mu(y)$ The second term is a non-negative function in x and y, since $\hat{k}^2$ positive definite on $(\mathbb{R}_+ \times \mathbb{R}_+) \times (\mathbb{R}_+ \times \mathbb{R}_+)$. Since $s(x,y)$ is also a non-negative function, the integration over $\mathcal{X} \times \mathcal{X}$ is positive. The independence of $K_{II}(P,Q)$ of the dominating measure follows from the one-homogeneity of $\hat{k}(x,y)$. Define now $f(x) = \hat{k}(p(x),q(x))$. Then $f \in L_1(\mathcal{X},\mu)$ since $\int_{\mathcal{X}} |f(x)|d\mu(x) \leq \int_{\mathcal{X}} \sqrt{\hat{k}(p(x),p(x))\hat{k}(q(x),q(x))}d\mu(x) = \kappa(\mathbb{R})^2 \int_{\mathcal{X}} \sqrt{p(x)q(x)}d\mu(x) \leq \kappa(\mathbb{R})^2$, where we have used the representation of one-homogeneous kernels. A bounded pd kernel $s(x,y)$ defines a positive definite integral operator $I : L_1(\mathcal{X},\mu) \to L_\infty(\mathcal{X},\mu)$, $(Ig)(x) = \int_{\mathcal{X}} s(x,y)g(y)d\mu(y)$. With the definition of $f(x)$ as above, K_{II} is positive since $K_{II}(P,Q) = \int_{\mathcal{X}} \int_{\mathcal{X}} s(x,y)f(x)f(y)d\mu(x)\mu(y) \geq 0$. □

Even if the kernel looks quite similar to the first one it cannot be decomposed as the first one, since $s(x,y)$ need not be a positive definite kernel. We just give the equivalent representation without proof:

Proposition 5.2 *If $s(x,y)$ is a positive definite kernel on $\mathcal{X}$, then $K_{II}(P,Q)$ can be equivalently written as:*

$$K_{II}(P,Q) = \int_T \left| \int_{\mathcal{X}} \Gamma(x,t)\hat{k}(p(x),q(x)d\mu(x) \right|^2 d\omega(t)$$

where $s(x,y) = \int_T \Gamma(x,t)\overline{\Gamma(x,t)}d\omega(t)$.

We illustrate this representation with a simple example. Let $s(x,y)$ be a translation-invariant kernel on $\mathbb{R}^n$. Then we can again use Bochner's theorem for the representation of $s(x,y)$. The proposition then states that the kernel $K_{II}(P,Q)$ is nothing else than the integrated power spectrum of the function $\hat{k}(p(x),q(x))$ with respect to ω.

6 Experiments

We compared the performance of the proposed metrics/kernels in four classification tasks. All used data sets consist of inherently positive data resp. counts of terms, counts of pixels of a given color, intensity at a given pixel. Also we will never encounter an infinite number of counts in practice, so that the assumption that the data consists of bounded, positive measures seems reasonable. Moreover we normalize always so that we get probability measures. For text data this is one of the standard representations, also for the Corel data this is quite natural, since all images have the same size and therefore the same number of pixels. This in turn implies that all images have the same mass in color space. For the USPS dataset it might seem at first a little bit odd to see digits as probability measures. Still the results we get are comparable to that of standard kernels without normalization, see [10]. Nevertheless we don't get state-of-the-art results for USPS since we don't implement invariance of the digits with respect to translations and small rotations. Details of the datasets and used similarity measures:

- *Reuters* text data set. The documents are represented as term histograms. Following [8] we used the five most frequent classes *earn, acq, moneyFx, grain* and *crude*. Documents which belong to more than one of theses classes are excluded. This results in a data set with 8085 examples of dimension 18635.
- *WebKB* web pages data set. The documents are also represented as term histograms. The four most frequent classes *student, faculty, course* and *project* are used. 4198 documents remain each of dimension 24212, see [8]. For both structural kernels we took for both text data sets the correlation matrix in the bag of documents representation as a pd kernel on the space of terms.
- *Corel* image data base. We chose the categories Corel14 from the Corel image database as in [1]. The Corel14 has 14 classes each with 100 examples. As reported in [1] the classes are very noisy, especially the bear and polar bear classes. We performed a uniform quantization of each image in the RGB color space, using 16 bins per color, yielding 4096 dimensional histograms. For both structural kernels we used as a similarity measure on the RGB color space, the compactly supported positive definite RBF kernel $k(x,y) = (1 - \|x-y\|/d_{max})_+^2$, with $d_{max} = 0.15$, see [14].
- *USPS* data set. 7291 training and 2007 test samples. For the first structural kernel we used again the compactly supported RBF kernel with $d_{max} = 2.2$, where we take the euclidean distance on the pixel space such that the smallest distance between two pixels is 1. For the second structural kernel we used as the similarity function $s(x,y) = 1_{\|x-y\| \leq 2.2}$.

All data sets were split into a training (80%) and a test (20%) set. The multi-class problem was solved by one-vs-all with SVM's. For all experiments we used the one-parameter family $d^2_{\alpha|1}$ of Hilbertian metrics resp. their positive definite kernel counterparts $k_{\alpha|1}$ as basic metrics resp. kernels on $\mathbb{R}_+$, in order to build the covariant Hilbertian metrics and both structural kernels. In the table they are denoted as *dir*. Then a second run was done by plugging the metric $D_{\alpha|1}(P,Q)$ on $\mathcal{M}^1_+(\mathcal{X})$ induced by the covariant resp. structural kernels into a Gaussian[9]:

$$K_{\alpha|1,\lambda}(P,Q) = e^{-D^2_{\alpha,1}(P,Q)/\lambda} \tag{9}$$

They are denoted in the table as *exp*. As a comparison we show the results if one takes the linear kernel on $\mathbb{R}_+$, $k(x,y) = xy$ as a basis kernel. Note that this kernel is 2-homogeneous compared to the 1-homogeneous kernels $k_{\alpha|1}$. Therefore the linear kernel will not yield a covariant kernel. As mentioned earlier the first structural kernel becomes independent of the dominating measure with this choice of $\hat{k}$. Also in this case we plugged the resulting metric on $\mathcal{M}^1_+(\mathcal{X})$ into a Gaussian for a second series of experiments. In the simplest case this gives the Gaussian kernel $k(x,y) = \exp(-\|x-y\|^2/\lambda)$.
For the penalty constant we chose from $C = \{10^k, k = -1,0,1,2,3,4\}$ and for α from $\alpha = \{1/2, \pm 1, \pm 2, \pm 4, \pm 16, \infty\}$ ($\alpha = -\infty$ coincides with $\alpha = \infty$). For the Gaussian (9) we chose additionally from $\lambda = 0.2 * \sigma * \{3,4,5,6,7,8,9,10,11,12,13\}$, where $\sigma = \frac{1}{n}\sum_{m=1}^n K(P_m, P_m)$. In order to find the best parameters for C, α resp. C, α, λ we performed 10-folds cross validation. For the best parameters among α, C resp. α, C, λ we evaluated the test error. Since the Hilbertian metrics of (5) were not yet compared or even used in kernel methods we also give the test errors for the kernels corresponding to $\alpha = -1, 1/2, 1, \infty$. The results are shown in table 1.

6.1 Interpretation

- The test error for the best α among the family $k_{\alpha|1}$ selected by cross-validation gives for all three types of kernels and their Gaussian transform always optimal or close to optimal results.

- For the text classification the covariant kernels were always better than the structured ones. We think that by using a better similarity measure on terms the structural kernels should improve. For the two image classification tasks the test errors of the best structural kernel is roughly 10% better than the best covariant one.

- The linear resp. Gaussian kernel were for the first three data-sets always worse than the corresponding covariant ones. This remains valid even if one only compares the direct covariant ones with the Gaussian kernel (so that one has in both cases only a one-parameter family of kernels). For the USPS dataset the results are comparable. Future experiments have to show whether this remains true if one considers unnormalized data.

[9] It is well-known that this transform yields a positive definite kernel iff D is a Hilbertian metric, see e.g. [2].

7 Conclusion

We went on with the work started in [5] on Hilbertian metrics resp. pd kernels on $\mathcal{M}^1_+(\mathcal{X})$. We extended a family of Hilbertian metrics proposed by Topsøe, so that now all previously used measures on probabilities are now included in this family. Moreover we studied further structural kernels on probability measures. We gave an equivalent representation for our first structural kernel on $\mathcal{M}^1_+(\mathcal{X})$, which on the one hand provides a better understanding how it captures structure of the probability measures and on the other hand gives in some cases a more efficient way to compute it. Further we proposed a second structural kernel which is independent of the dominating measure, therefore yielding a structural kernel on all probability measures. Finally we could show that doing model selection in $d^2_{\alpha|1}$ resp. $k_{\alpha|1}$ gives almost optimal results for covariant and structural kernels. Also the covariant kernels and their Gaussian transform are almost always superior to the linear resp. the Gaussian kernel, which suggests that the considered family of kernels is a serious alternative whenever one has data which is generically positive. It remains an open problem if one can improve the structural kernels for text classification by using a better similarity function/kernel.

Acknowledgements

We would like to thank Guy Lebanon for kindly providing us with the WebKB and Reuters data set in preprocessed from. Furthermore we are thankful to Flemming Topsøe and Bent Fuglede for providing us with preprints of their papers [13, 3].

References

[1] O. Chapelle, P. Haffner, and V. Vapnik. SVMs for histogram-based image classification. *IEEE Trans. on Neural Networks*, 10:1055–1064, 1999.

[2] J. P. R. Christensen C. Berg and P. Ressel. *Harmonic Analysis on Semigroups*. Springer, New York, 1984.

[3] B. Fuglede. Spirals in Hilbert space. With an application in information theory. To appear in Expositiones Mathematicae, 2004.

Table 1: The table shows the test errors for the covariant and the two structural kernels resp. of their Gaussian transform for each data set. The first column shows the test error and the α-value of the kernel with the best cross-validation error over the family $D^2_{\alpha|1}$ denoted as *dir* resp. of the Gaussian transform denoted as *exp*. The next four columns provide the results for the special cases $\alpha = -1, 1/2, 1, \infty$ in $D^2_{\alpha|1}$ resp. $K_{\alpha|1,\lambda}$. The last column $\langle \cdot, \cdot \rangle$ gives the test error if one takes the linear kernel as basis kernel resp. of the Gaussian transform.

			Best α		$\alpha = -1$	$\alpha = \frac{1}{2}$	$\alpha = 1$	$\alpha = \infty$	$\langle \cdot, \cdot \rangle$
Reuters	cov	dir	**1.36**	-1	**1.36**	1.42	**1.36**	1.79	1.98
	cov	exp	*1.54*	1/2	1.73	1.54	1.79	1.91	1.73
	str	dir	1.85	1	*1.60*	1.91	1.85	1.67	2.16
	str	exp	*1.54*	1	1.60	*1.54*	*1.54*	1.60	2.10
	str2	dir	*1.54*	1	1.85	1.67	*1.54*	2.35	2.41
	str2	exp	*1.67*	1/2	2.04	1.67	1.91	2.53	2.65
WebKB	cov	dir	4.88	16	4.76	4.88	*4.52*	4.64	7.49
	cov	exp	4.76	1	4.76	**4.40**	4.76	4.99	7.25
	str	dir	*4.88*	∞	5.47	5.95	5.23	*4.88*	6.30
	str	exp	*5.11*	∞	5.35	5.23	*5.11*	*5.11*	6.42
	str2	dir	*4.88*	1/2	5.59	*4.88*	5.59	6.30	9.39
	str2	exp	*5.59*	1/2	6.18	5.59	5.95	7.13	9.04
Corel14	cov	dir	12.86	-1	12.86	20.71	15.71	*12.50*	30.00
	cov	exp	12.50	1	*11.43*	14.29	12.50	11.79	34.64
	str	dir	15.71	-1	15.71	23.21	16.43	*12.14*	29.64
	str	exp	**10.36**	1	10.71	12.50	**10.36**	11.07	20.36
	str2	dir	20.00	16	*18.57*	21.43	19.29	20.00	36.79
	str2	exp	*17.14*	1/2	18.57	*17.14*	19.29	18.93	35.71
USPS	cov	dir	*7.82*	-2	8.07	7.92	8.17	7.87	9.02
	cov	exp	*4.53*	-16	4.58	4.58	*4.53*	5.28	*4.53*
	str	dir	*7.52*	-1	*7.52*	8.87	7.77	7.87	9.07
	str	exp	4.04	1/2	3.99	4.04	**3.94**	4.78	4.09
	str2	dir	5.48	2	5.18	5.28	5.33	6.03	*5.03*
	str2	exp	4.29	1/2	*4.09*	4.29	4.24	5.03	4.88

[4] M. Hein, O. Bousquet, and B. Schölkopf. Maximal margin classification for metric spaces. *Journal of Computer and System Sciences*, to appear.

[5] M. Hein, T. N. Lal, and O. Bousquet. Hilbertian metrics on probability measures and their application in SVM's. In *26th Pattern Recognition Symposium (DAGM)*. Springer, 2004.

[6] S. Janson. *Gaussian Hilbert Spaces*. Cambridge University Press, Cambridge, 1997.

[7] T. Jebara and R. Kondor. Bhattacharyya and expected likelihood kernels. In *16th Annual Conference on Learning Theory (COLT)*, 2003.

[8] J. Lafferty and G. Lebanon. Diffusion kernels on statistical manifolds. Technical Report CMU-CS-04-101, School of Computer Science, Carnegie Mellon University, Pittsburgh, 2004.

[9] P. J. Moreno, P. P. Hu, and N. Vasconcelos. A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. *NIPS*, 16, 2003.

[10] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[11] B. Schölkopf. The kernel trick for distances. *NIPS*, 13, 2000.

[12] C. Suquet. Distances euclidiennes sur les mesures signées et application à des théorèmes de Berry-Esséen. *Bull. Belg. Math. Soc. Simon Stevin*, 2:161–181, 1995.

[13] F. Topsøe. Jenson-Shannon divergence and norm-based measures of discrimination and variation. Preprint, 2003.

[14] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial basis functions of minimal degree. *Adv. Comp. Math.*, 4:389–396, 1995.

Fast Non-Parametric Bayesian Inference on Infinite Trees

Marcus Hutter
IDSIA, Galleria 2, CH-6928 Manno-Lugano, Switzerland
marcus@idsia.ch http://www.idsia.ch/~marcus

Abstract

Given i.i.d. data from an unknown distribution, we consider the problem of predicting future items. An adaptive way to estimate the probability density is to recursively subdivide the domain to an appropriate data-dependent granularity. A Bayesian would assign a data-independent prior probability to "subdivide", which leads to a prior over infinite(ly many) trees. We derive an exact, fast, and simple inference algorithm for such a prior, for the data evidence, the predictive distribution, the effective model dimension, and other quantities.

1 INTRODUCTION

Inference. We consider the problem of inference from i.i.d. data D, in particular of the unknown distribution q the data is sampled from. In case of a continuous domain this means inferring a probability density from data. Without structural assumption on q, this is hard to impossible, since a finite amount of data is never sufficient to uniquely select a density (model) from an infinite-dimensional space of densities (model class).

Methods. In parametric estimation one assumes that q belongs to a finite-dimensional family. The two-dimensional family of Gaussians characterized by mean and variance is prototypical. The maximum likelihood (ML) estimate of q is the distribution that maximizes the data likelihood. Maximum likelihood overfits if the family is too large and especially if it is infinite-dimensional. A remedy is to penalize complex distributions by assigning a prior (2nd order) probability to the densities q. Maximizing the model posterior (MAP), which is proportional to likelihood times the prior, prevents overfitting. Bayesians keep the complete posterior for inference. Typically, summaries like the mean and variance of the posterior are reported.

How to choose the prior? In finite or small compact low-dimensional spaces a uniform prior often works (MAP reduces to ML). In the non-parametric case one typically devises a hierarchy of finite-dimensional model classes of increasing dimension. Selecting the dimension with maximal posterior often works well due to the Bayes factor phenomenon [Goo83, Jay03, Mac03]: In case the true model is low-dimensional, higher-dimensional (complex) model classes are automatically penalized, since they contain fewer "good" models. Full Bayesians would assign a prior probability (e.g. $\frac{1}{d^2}$) to dimension d and mix over dimension.

Interval Bins. The probably simplest and oldest model for an interval domain is to divide the interval (uniformly) into bins, assume a constant distribution within each bin, and take a frequency estimate for the probability in each bin, or a Dirichlet posterior if you are a Bayesian. There are heuristics for choosing the number of bins as a function of the data size. The simplicity and easy computability of the bin model is very appealing to practitioners. Drawbacks are that distributions are discontinuous, its restriction to one dimension (or at most low dimension: curse of dimensionality), the uniform (or more generally fixed) discretization, and the heuristic choice of the number of bins. We present a full Bayesian solution to these problems, except for the non-continuity problem. Polya trees [Lav94] inspired our model.

More advanced model classes. There are plenty of alternative Bayesian models that overcome some or all of the limitations. Examples are continuous Dirichlet process (mixtures) [Fer73], Bernstein polynomials [PW02], Bayesian field theory [Lem03], Bayesian kernel density estimation or other mixture models [EW95], or universal priors [Hut04b], but analytical solutions are infeasible. Markov Chain Monte Carlo sampling or Expectation Maximization algorithms [DLR77] or variational methods can often be used to obtain approximate numerical solutions, but computation time and global convergence remain crit-

ical issues. Practitioners usually use (with success) efficient MAP or M(D)L or heuristic methods, e.g. kernel density estimation [GM03], but note that MAP or MDL *can* fail, while Bayes works [PH04].

Our tree mixture model. The idea of the model class discussed in this paper is very simple: With equal probability, we chose q either uniform or split the domain in two parts (of equal volume), and assign a prior to each part, recursively, i.e. in each part again either uniform or split. For finitely many splits, q is a piecewise constant function, for infinitely many splits it is virtually *any* distribution. While the prior over q is neutral about uniform versus split, we will see that the posterior favors a split if and only if the data clearly indicates non-uniformity. The method is a full Bayesian non-heuristic tree approach to adaptive binning for which we present a very simple and fast algorithm for computing all(?) quantities of interest.

Contents. In Section 2 we introduce our model and compare it to Polya trees. We also discuss some example domains, like intervals, strings, volumes, and classification tasks. In Section 3 we present recursions for various quantities of interest, including the data evidence, the predictive distribution, the effective model dimension, the tree size and height, and cell volume. We discuss the qualitative behavior and state convergence of the posterior for finite trees. The proper case of infinite trees is discussed in Section 4, where we analytically solve the infinite recursion at the data separation level. Section 5 collects everything together and presents the algorithm. We also numerically illustrate the behavior of our model on one example distribution. Section 6 contains a brief summary, conclusions, and outlook, including natural generalizations of our model. See [Hut04a] for derivations, proofs, program code, extensions, and more details.

2 THE TREE MIXTURE MODEL

Setup and basic quantities of interest. We are given i.i.d. data $D=(x^1,...,x^n)\in\Gamma^n$ of size n from domain Γ, e.g. $\Gamma\subseteq\mathbb{R}^d$ sampled from some unknown probability density $q:\Gamma\to\mathbb{R}$. Standard inference problems are to estimate q from D or to predict the next data item $x^{n+1}\in\Gamma$. By definition, the (objective or aleatoric) data likelihood density under model q is $p(D|q)\equiv q(x_1)\cdot...\cdot q(x_n)$. Note that we consider sorted data, which avoids annoying multinomial coefficients. Otherwise this has no consequences. Results are independent of the order and depend on the counts only, as they should. A Bayesian assumes a (belief or 2^{nd}-order or epistemic or subjective) prior $p(q)$ over models q in some model class Q. The data evidence is $p(D)=\int_Q p(D|q)p(q)dq$. Having the evidence, Bayes' famous rule allows to compute the (belief or 2^{nd}-order or epistemic or subjective) posterior $p(q|D)=p(D|q)p(q)/p(D)$ of q. The predictive or posterior distribution of x is $p(x|D)=p(D,x)/p(D)$, i.e. the conditional probability that the next data item is $x=x^{n+1}$, given D, follows from the evidences of D and (D,x). Since the posterior of q is a complex object, we need summaries like the expected q-probability of x and (co)variances. Fortunately they can also be reduced to computation of evidences: $E[q(x)|D]:=\int q(x)p(q|D)dq=p(x|D)$. In the last equality we used the formulas for the posterior, the likelihood, the evidence, and the predictive distribution, in this order. Similarly for the covariance. We derive and discuss further summaries of q for our particular tree model, like the model complexity or effective dimension, and the tree height or cell size, later.

Hierarchical tree partitioning. Up to now everything has been fairly general. We now introduce the tree representation of domain Γ. We partition Γ into Γ_0 and Γ_1, i.e. $\Gamma=\Gamma_0\cup\Gamma_1$ and $\Gamma_0\cap\Gamma_1=\phi$. Recursively we (sub)partition $\Gamma_z=\Gamma_{z0}\dot\cup\Gamma_{z1}$ for $z\in\mathbb{B}_0^m$, where $\mathbb{B}_k^m=\bigcup_{i=k}^m\{0,1\}^i$ is the set of all binary strings of length between k and m, and $\Gamma_\epsilon=\Gamma$, where $\epsilon=\{0,1\}^0$ is the empty string. We are interested in an infinite recursion, but for convenience we assume a finite tree height $m<\infty$ and consider $m\to\infty$ later. Also let $l:=\ell(z)$ be the length of string $z=z_1...z_l=:z_{1:l}$, and $|\Gamma_z|$ the volume or length or cardinality of Γ_z.

Example spaces. *Intervals:* Assume $\Gamma=[0,1)$ is the unit interval, recursively bisected into intervals $\Gamma_z=[0.z,0.z+2^{-l})$ of length $|\Gamma_z|=2^{-l}$, where $0.z$ is the real number in $[0,1)$ with binary expansion $z_1...z_l$.

Strings: Assume $\Gamma_z=\{zy:y\in\{0,1\}^{m-l}\}$ is the set of strings of length m starting with z. Then $\Gamma=\{0,1\}^m$ and $|\Gamma_z|=2^{m-l}$. For $m=\infty$ this set is continuous, for $m<\infty$ finite.

Trees: Let Γ be a complete binary tree of height m and Γ_{z0} (Γ_{z1}) be the left (right) subtree of Γ_z. If $|\Gamma_z|$ is defined as one more than the number of nodes in Γ_z, then $|\Gamma_z|=2^{m+1-l}$.

Volumes: Consider $\Gamma\subset\mathbb{R}^d$, e.g. the hypercube $\Gamma=[0,1)^d$. We recursively halve Γ_z with a hyperplane orthogonal to dimension $(l \bmod d)+1$, i.e. we sweep through all orthogonal directions. $|\Gamma_z|=2^{-l}|\Gamma|$.

Compactification: We can compactify $\Gamma\subseteq(1,\infty]$ (this includes $\Gamma=\mathbb{N}\setminus\{1\}$) to the unit interval $\Gamma':=\{\frac{1}{x}:x\in\Gamma\}\subseteq[0,1)$, and similarly $\Gamma\subseteq\mathbb{R}$ (this includes $\Gamma=\mathbb{Z}$) to $\Gamma':=\{x\in[0,1):\frac{2x-1}{x(1-x)}\in\Gamma\}$. All reasonable spaces can be reduced to one of the spaces described above.

Classification: Consider an observation $o\in\Gamma'$ (e.g. email) that is classified as $c\in\{0,1\}$ (e.g. good versus spam), where Γ' could be one of the spaces above (e.g. o is a sequence of binary features in decreasing order

of importance). Then $x := (o,c) \in \Gamma := \Gamma' \times \{0,1\}$ and $\Gamma_{0z} = \Gamma'_z \times \{0\}$ and $\Gamma_{1z} = \Gamma'_z \times \{1\}$. Given D (e.g. preclassified emails), a new observation o is classified as c with probability $p(c|D,o) \propto p(D,x)$. Similar for more than two classes.

In all these examples (we have chosen) $|\Gamma_{z0}| = |\Gamma_{z1}| = \frac{1}{2}|\Gamma_z| \; \forall z \in \mathbb{B}_0^{m-1}$, and this is the only property we need and henceforth assume. W.l.g. we assume/define/rescale $|\Gamma| = 1$. Generalizations to non-binary and non-symmetric partitions are straightforward and briefly discussed at the end.

Identification. We assume that $\{\Gamma_z : z \in \mathbb{B}_0^m\}$ are (basis) events that generate our σ-algebra. For every $x \in \Gamma$ let x' be the string of length $\ell(x') = m$ such that $x \in \Gamma_{x'}$. We assume that distributions q are σ-measurable, i.e. to be constant on $\Gamma_{x'} \; \forall x' \in \mathbb{B}^m$. For $m = \infty$ this assumption is vacuous; we get *all* Borel measures. Hence, we can identify the continuous sample space Γ with the (for $m < \infty$ discrete) space $\mathbb{B}^m$ of binary sequences of length m, i.e. in a sense all example spaces are isomorphic. While we have the volume model in mind for real-world applications, the string model will be convenient for mathematical notation, the tree metaphor will be convenient in discussion, and the interval model will be easiest to implement and to present graphically.

Notation. As described above, Γ may also be a tree. This interpretation suggests the following scheme for defining the probability of q on the leaves x'. The probability of the left child node $z0$, given we are in the parent node z, is $P[\Gamma_{z0}|\Gamma_z,q]$, so we have

$$p(x|\Gamma_z,q) = p(x|\Gamma_{z0},q) \cdot P[\Gamma_{z0}|\Gamma_z,q] \quad \text{if} \quad x \in \Gamma_{z0}$$

and similarly for the right child. In the following we often have to consider distributions conditioned to and in the subtree Γ_z, so the following notation will turn out convenient

$$q_{z0} := P[\Gamma_{z0}|\Gamma_z,q], \quad p_z(x|...) := 2^{-l} p(x|\Gamma_z...) \quad (1)$$

$$\Rightarrow p_z(x|q) = 2q_{zx_{l+1}} p_{zx_{l+1}}(x|q) = ... = \prod_{i=l+1}^{m} 2q_{x_{1:i}} \text{ if } x \in \Gamma_z$$

where we have used that $p(x|\Gamma_{x'},q) = |\Gamma_{x'}|^{-1} = 2^m$ is uniform. Note that $q_{z0} + q_{z1} = 1$. Finally, let $\vec{q}_{z*} := (q_{zy} : y \in \mathbb{B}_1^{m-l})$ be the $(2^{m-l+1}-2)$-dimensional *vector* or *ordered set* or *tree* of all *reals* $q_{zy} \in [0,1]$ in subtree Γ_z. Note that $q_z \notin \vec{q}_{z*}$. The *(non)density* $q_z(x) := p_z(x|q)$ depends on all and only these q_{zy}. For $z \neq \epsilon$, $q_z()$ and $p_z()$ are only proportional to a density due to the factor 2^{-l}, which has been introduced to make $p_{x'}(x|...) \equiv 1$. (They are densities w.r.t. $2^l \lambda_{|\Gamma_z}$, where λ is the Lebesgue measure.) We have to keep this in mind in our derivations, but can ignore this widely in our discussion.

Polya trees. In the Polya tree model one assumes that the $q_{z0} \equiv 1 - q_{z1}$ are independent and Beta$(\cdot,\cdot)$ distributed, which defines the prior over q. Polya trees form a conjugate prior class, since the posterior is also a Polya tree, with empirical counts added to the Beta parameters. If the same Beta is chosen in each node, the posterior of x is pathological for $m \to \infty$: The distribution is everywhere discontinuous with probability 1. A cure is to increase the Beta parameters with l, e.g. quadratically, but this results in "underfitting" for large sample sizes, since Beta(large,large) is too informative and strongly favors q_{z0} near $\frac{1}{2}$. It also violates scale invariance, which should hold in the absence of prior knowledge. That is, the p(oste)rior in $\Gamma_0 = [0,\frac{1}{2})$ should be the same as for $\Gamma = [0,1)$ (after rescaling all $x \rightsquigarrow x/2$ in D).

The new tree mixture model. The prior $P[q]$ follows from specifying a prior over $\vec{q}_*$, since $q(x) \propto q_{x_1} \cdot ... \cdot q_{x_{1:m}}$ by (1). The distribution in each subset $\Gamma_z \subseteq \Gamma$ shall be either *u*niform or non-uniform. A necessary (but not sufficient) condition for uniformity is $q_{z0} = q_{z1} = \frac{1}{2}$.

$$p^u(q_{z0}, q_{z1}) \;:=\; \delta(q_{z0} - \tfrac{1}{2})\delta(q_{z1} - \tfrac{1}{2}), \quad (2)$$

where $\delta()$ is the Dirac delta. To get uniformity on Γ_z we have to recurse the tree down in this way.

$$p^u(\vec{q}_{z*}) \;:=\; p^u(q_{z0}, q_{z1}) p^u(\vec{q}_{z0*}) p^u(\vec{q}_{z1*}) \quad (3)$$

with the natural recursion termination $p^u(\vec{q}_{z*}) = 1$ when $\ell(z) = m$, since then $\vec{q}_{z*} = \phi$. For a non-uniform distribution on Γ_z we allow any probability split $q(\Gamma_z) = q(\Gamma_{z0}) + q(\Gamma_{z0})$, or equivalently $1 = q_{z0} + q_{z1}$. We assume a uniform prior on the *s*plit, i.e.

$$p^s(q_{z0}, q_{z1}) \;:=\; \delta(q_{z0} + q_{z1} - 1) \quad (4)$$

We now recurse down the tree

$$p^s(\vec{q}_{z*}) \;:=\; p^s(q_{z0}, q_{z1}) p(\vec{q}_{z0*}) p(\vec{q}_{z1*}) \quad (5)$$

again with the natural recursion termination $p(\vec{q}_{z*}) = p(\phi) = 1$ when $\ell(z) = m$. Finally we have to mix the uniform with the non-uniform case.

$$p(\vec{q}_{z*}) \;:=\; p(u) p^u(\vec{q}_{z*}) + p(s) p^s(\vec{q}_{z*}) \quad (6)$$

We choose a 50/50 mixture $p(u) = p(s) = \frac{1}{2}$. This completes the specification of the prior $P[q] = p(\vec{q}_*)$.

For example, if the first bit in x is a class label and the remaining are binary features in decreasing order of importance, then given class and features $z = x_{1:l}$, further features $x_{l+1:m}$ could be relevant for classification ($q_z(x)$ is non-uniform) or irrelevant ($q_z(x)$ is uniform).

Comparison to the Polya tree. Note the important difference in the recursions (3) and (5). Once we decided on a uniform distribution (2) we have to equally

split probabilities down the recursion to the end, i.e. we recurse in (3) with p^u, rather than the mixture p (this actually allows to solve the recursion). On the other hand if we decided on a non-uniform split (4), the left and right partition each itself may be uniform or not, i.e. we recurse in (5) with the mixture p, rather than p^s. Inserting (4) in (5) in (6) and recursively (2) in (3) in (6) we get

$$p(\vec{q}_{z*}) = \tfrac{1}{2}\prod_{y\in\mathbb{B}_1^{m-l}}\delta(q_{zy}-\tfrac{1}{2}) + \tfrac{1}{2}\delta(q_{z0}+q_{z1}-1)p(\vec{q}_{z0*})p(\vec{q}_{z1*}) \quad (7)$$

Choosing $p(u)=0$ would lead to the Polya tree model (and its problems) with $q_{z0}\sim$ Beta(1,1). For our choice ($p(u)=\frac{1}{2}$), but with p instead of p^u on the r.h.s. of (3) we would get a quasi-Polya model (same problems) with $q_{z0}\sim\frac{1}{2}$[Beta(∞,∞)+Beta(1,1)].

For $m\to\infty$, our model is scale invariant *and* leads to continuous distributions for $n\to\infty$, unlike the Polya tree model. We also don't have to tune Beta parameters; the model tunes itself by suitably assigning high/low posterior probability to subdividing cells. While Polya trees form a natural conjugate prior class, our prior does not directly, but can be generalized to do so [Hut04a]. The computational complexity for the quantities of interest will be the same (essentially $O(n)$), i.e. as good as it could be.

Formal and effective dimension. Formally our model is $2\cdot(2^m-1)$-dimensional, but the effective dimension can by much smaller, since $\vec{q}_*$ is forced with a non-zero probability to a much smaller polytope, for instance with probability $\frac{1}{2}$ to the zero-dimensional globally uniform distribution. We will compute the effective p(oste)rior dimension.

3 QUANTITIES OF INTEREST

The evidence recursion. At the end of Section 2 we defined our tree mixture model. The next step is to compute the standard quantities of interest defined at the beginning of Section 2. The evidence $p(D)$ is key, the other quantities (posterior, predictive distribution, expected $q(x)$ and its variance) follow then immediately. Let $D_z:=\{x\in D: x\in\Gamma_z\}$ be the $n_z:=|D_z|$ data points that lie in subtree Γ_z. We compute $p_z(D_z)$ recursively for all $z\in\mathbb{B}_0^{m-1}$, which gives $p(D)=p_\epsilon(D_\epsilon)$. Inserting (1) and (7) into

$$p_z(D_z) = \int p_z(D_z|\vec{q}_{z*})p(\vec{q}_{z*})d\vec{q}_{z*} \quad (8)$$

one can derive the following recursion [Hut04a]:

$$\begin{aligned} p_z(D_z) &= \tfrac{1}{2}\big[1+\frac{p_{z0}(D_{z0})p_{z1}(D_{z1})}{w(n_{z0},n_{z1})}\big] \quad (9)\\ w(n_{z0},n_{z1}) &:= 2^{-n_z}\frac{(n_z+1)!}{n_{z0}!n_{z1}!} =: w_{n_z}(\Delta_z)\\ n_z &= n_{z0}+n_{z1}, \quad \Delta_z := \frac{n_{z0}}{n_z}-\tfrac{1}{2}\end{aligned}$$

The recursion terminates with $p_z(D_z)=1$ when $\ell(z)=m$. Recall (1) if you insist on a formal proof: For $\ell(z)=m$ and $x\in\Gamma_z$ we have $\Gamma_{x'}=\Gamma_z \Rightarrow p_z(x|q)=1 \Rightarrow p_z(D_z|q)=1 \Rightarrow p_z(D_z)=1$.

Interpretation of (9): With probability $\frac{1}{2}$, the evidence is uniform in Γ_z. Otherwise data D_z is split into two partitions of size n_{z0} and $n_{z1}=n_z-n_{z0}$. First, choose n_{z0} uniformly in $\{0,...,n_z\}$. Second, given n_z, choose uniformly among the $\binom{n_z}{n_{z0}}$ possibilities of selecting n_{z0} out of n_z data points for Γ_{z0} (the remaining n_{z1} are then in Γ_{z1}). Third, distribute D_{z0} according to $p_{z0}(D_{z0})$ and D_{z1} according to $p_{z1}(D_{z1})$. Then, the evidence in case of a split is the second term in (9). The factor 2^{n_z} is due to our normalization convention (1). This also verifies that the r.h.s. yields the l.h.s. if integrated over all D_z, as it should be.

Discussing the weight. The relative probability of splitting (second term on r.h.s. of (9)) to the uniform case (first term in r.h.s. of (9)) is controlled by the weight w. Large (small) weight indicates a (non) uniform distribution, provided p_{z0} and p_{z1} are $O(1)$. Balance $\Delta_z\approx 0$ ($\not\approx 0$) indicates a (non) symmetric partitioning of the data among the left and right branch of Γ_z. Asymptotically for large n_z (keeping Δ_z fixed), we have

$$w_{n_z}(\Delta_z) \sim \sqrt{\tfrac{2n_z}{\pi}}\,\mathrm{e}^{-2n_z\Delta_z^2}$$

Assume that data D is sampled from the true distribution $\dot{q}$. The probability of the left branch Γ_{z0} of Γ_z is $\dot{q}_{z0}\equiv P[\Gamma_{z0}|\Gamma_z,\dot{q}]=2^l\dot{q}_z(\Gamma_{z0})$. The relative frequencies $\frac{n_{z0}}{n_z}$ asymptotically converge to $\dot{q}_{z0}$. More precisely $\frac{n_{z0}}{n_z}=\dot{q}_{z0}\pm O(n_z^{-1/2})$ with probability 1 (w.p.1). Similarly for the right branch. Assume the probabilities are equal ($\dot{q}_{z0}=\dot{q}_{z1}=\frac{1}{2}$), possibly but not necessarily due to a uniform $\dot{q}_z()$ on Γ_z. Then $\Delta_z=O(n_z^{-1/2})$, which implies

$$w_{n_z}(\Delta_z)\sim\Theta(\sqrt{n_z}) \overset{n_z\to\infty}{\underset{w.p.1}{\longrightarrow}} \infty \quad \text{if} \quad \dot{q}_{z0}=\dot{q}_{z1}=\tfrac{1}{2},$$

consistent with our anticipation. Conversely, for $\dot{q}_{z0}\neq\dot{q}_{z1}$ (which implies non-uniformity of $\dot{q}_z()$) we have $\Delta_z\to c:=\dot{q}_{z0}-\frac{1}{2}\neq 0$, which implies

$$w_{n_z}(\Delta_z)\sim\sqrt{\tfrac{2n_z}{\pi}}\,\mathrm{e}^{-2n_zc^2} \overset{n_z\to\infty}{\underset{w.p.1}{\longrightarrow}} 0 \quad \text{if} \quad \dot{q}_{z0}\neq\dot{q}_{z1},$$

again, consistent with our anticipation.

Asymptotic convergence/consistency ($n\to\infty$). For fixed $m<\infty$, one can show that almost surely the posterior $p_z(\vec{q}_{z*}|D)$ concentrates around the true distribution $\dot{\vec{q}}_{z*}$ for $n\to\infty$. This implies that the posterior $p_z(x|D_z)\to\dot{q}_z(x)$ for all $x\in\Gamma_z$. One can also show that the evidence $p_z(D_z)\to\frac{1}{2}$ or 1 for uniform $\dot{q}_z()$, and increases exponentially with n_z for non-uniform $\dot{q}_z()$ (see [Hut04a] for proofs).

Model dimension and cell number. As discussed in Section 2, the effective dimension of $\vec{q}_*$ is the number

of components that are not forced to $\frac{1}{2}$ by (2). Note that a component may be "accidentally" $\frac{1}{2}$ in (4), but since this is an event of probability 0, we don't have to care about this subtlety. So the effective dimension $N_{\vec{q}_{z*}}=\#\{q\in\vec{q}_{z*}:q\neq\frac{1}{2}\}$ of $\vec{q}_{z*}$ can be given recursively as

$$N_{\vec{q}_{z*}} \;=\; \begin{cases} 0 & \text{if} \quad \ell(z)=m \quad \text{or} \quad q_{z0}=\frac{1}{2} \\ 1+N_{\vec{q}_{z0*}}+N_{\vec{q}_{z1*}} & \text{else} \end{cases} \tag{10}$$

The effective dimension is zero if $q_z=\frac{1}{2}$, since this implies that the whole tree Γ_z has $q_{zy}=\frac{1}{2}$ due to (7). If $q_z\neq\frac{1}{2}$, we add the effective dimensions of subtrees Γ_{z0} and Γ_{z1} to the root degree of freedom $q_{z0}=q_z-q_{z1}$. Bayes' rule allows to represent the posterior probability that $N_{\vec{q}_{z*}}=k$ as

$$P_z[N_{\vec{q}_{z*}}=k|D_z]\cdot p_z(D_z)=\int\delta_{N_{\vec{q}_{z*}}k}p_z(D_z|\vec{q}_{z*})p(\vec{q}_{z*})d\vec{q}_{z*}$$

where $P_z[...|...]:=P[...|\Gamma_z...]$, and $\delta_{ab}=1$ for $a=b$ and 0 else. The r.h.s. coincides with (8) except for the extra factor $\delta_{N_{\vec{q}_{z*}}k}$. Analogous to the evidence (8), using (10) we can prove the following recursion:

$$\begin{aligned} &P_z[N_{\vec{q}_{z*}}=0|D_z] \;=\; 1-g_z(D_z), \qquad\qquad \text{for} \quad l<m, \\ &P_z[N_{\vec{q}_{z*}}\underset{k}{=}k+1|D_z] \;= \qquad\qquad\qquad\qquad\qquad\qquad (11) \\ &g_z(D_z)\cdot\sum_{i=0}P_{z0}[N_{\vec{q}_{z0*}}=i|D_{z0}]\cdot P_{z1}[N_{\vec{q}_{z1*}}=k-i|D_{z1}], \\ &P_z[N_{\vec{q}_{z*}}=k|D_z]=\delta_{k0}:=\left\{\begin{smallmatrix}1 \text{ if } k=0\\ 0 \text{ if } k>0\end{smallmatrix}\right\} \quad \text{for} \quad l=m. \\ &g_z(D_z):=\tfrac{1}{2}\frac{p_{z0}(D_{z0})p_{z1}(D_{z1})}{p_z(D_z)w(n_{z0},n_{z1})}\overset{(9)}{=}1-\frac{1}{2p_z(D_z)} \qquad (12) \end{aligned}$$

Read: The probability that tree Γ_z has dimension $k+1$ equals the posterior probability $g_z(D_z)$ of splitting Γ_z, times the probability that left subtree has dimension i, times the probability that right subtree has dimension $k-i$, summed over all possible i.

Let us define a cell or bin as a maximal volume on which $q()$ is constant. Then the model dimension is 1 less than the number of bins (due to the probability constraint). Hence we also have a recursion for the distribution of the number of cells.

Tree height and cell size. The effective height of tree $\vec{q}_{z*}$ at $x\in\Gamma_z$ is also an interesting property. If $q_{z0}=\frac{1}{2}$ or $\ell(z)=m$, then the height $h_{\vec{q}_{z*}}(x)$ of tree $\vec{q}_{z*}$ at x is obviously zero. If $q_{z0}\neq\frac{1}{2}$, we take the height of the subtree $\vec{q}_{zx_{l+1}*}$ that contains x and add 1:

$$h_{\vec{q}_{z*}}(x) \;=\; \begin{cases} 0 & \text{if} \quad \ell(z)=m \quad \text{or} \quad q_{z0}=\frac{1}{2} \\ 1+h_{\vec{q}_{zx_{l+1}*}}(x) & \text{else} \end{cases}$$

One can show that the tree height at x averaged over all trees $\vec{q}_{z*}$ is

$$E_z[h_{\vec{q}_{z*}}(x)|D_z]=g_z(D_z)\Big[1+E_{zx_{l+1}}[h_{\vec{q}_{zx_{l+1}*}}(x)|D_{zx_{l+1}}]\Big]$$

where $E_z[f_{\vec{q}_{z*}}|...]=\int P_z[f_{\vec{q}_{z*}}|...]p(\vec{q}_{z*})d\vec{q}_{z*}$. We may also want to compute the tree height averaged over all $x\in\Gamma_z$. For $\ell(z)<m$ and $q_{z0}\neq\frac{1}{2}$ we get

$$\bar{h}_{\vec{q}_{z*}} \;:=\; \int h_{\vec{q}_{z*}}(x)q(x|\Gamma_z)dx \;=\; 1+q_{z0}\cdot\bar{h}_{\vec{q}_{z0*}}+q_{z1}\cdot\bar{h}_{\vec{q}_{z1*}}$$

$$\begin{aligned} E_z[\bar{h}_{\vec{q}_{z*}}|D_z] \;=\; g_z(D_z)\Big[1+\frac{n_{z0}+1}{n_z+2}E_{z0}[\bar{h}_{\vec{q}_{z0*}}|D_{z0}] \\ +\frac{n_{z1}+1}{n_z+2}E_{z1}[\bar{h}_{\vec{q}_{z1*}}|D_{z1}]\Big] \end{aligned}$$

with obvious interpretation: The expected height of a subtree is weighted by its relative importance, that is (an estimate of) its probability. The recursion terminates with $E_z[h_{\vec{q}_{z*}}|D_z]=0$ when $\ell(z)=m$. We can also compute intra and inter tree height variances.

Finally consider the average cell size or volume v. Maybe more useful is to consider the logarithm $-\log_2|\Gamma_z|=\ell(z)$, since otherwise small volumes can get swamped in the expectation by a single large one. Log-volume $v_{\vec{q}_{z*}}=\ell(z)$ if $\ell(z)=m$ or $q_z=\frac{1}{2}$, and else recursively $v_{\vec{q}_{z*}}=q_{z0}v_{\vec{q}_{z0*}}+q_{z1}v_{\vec{q}_{z1*}}$. We can reduce this to the tree height, since $v_{\vec{q}_{z*}}=\bar{h}_{\vec{q}_{z*}}+\ell(z)$, in particular $v_{\vec{q}_*}=\bar{h}_{\vec{q}_*}$

4 INFINITE TREES ($m\to\infty$)

Motivation. We have chosen an (arbitrary) finite tree height m in our setup, needed to have a well-defined recursion start at the leaves of the trees. What we are really interested in are infinite trees ($m=\infty$). Why not feel lucky with finite m? First, for continuous domain Γ (e.g. interval [0,1)), our tree model contains only piecewise constant models. The true distribution $\dot{q}()$ is typically non-constant and continuous (Beta, normal, ...). Such distributions are outside a finite tree model class (but inside the infinite model), and the posterior $p(x|D)$ cannot converge to the true distribution, since it is also piecewise constant. Hence all other estimators based on the posterior are also not consistent. Second, a finite m violates scale invariance (a non-informative prior on Γ_z should be the same for all z, apart from scaling). Finally, having to choose the "right" m may be worrisome.

For increasing m, the cells Γ_x become smaller and will (normally) eventually contain either only a single data item, or be empty. It should not matter whether we further subdivide empty or singleton cells. So we expect inferences to be independent of m for sufficiently large m, or at least the limit $m\to\infty$ to exist. In this section we show that this is essentially true.

Prior inferences ($D=\phi$). We first consider the prior (zero data) case $D=\phi$. Recall that $z\in\mathbb{B}_0^m$ is some node and $x\in\mathbb{B}^m$ a leaf node. Normalization

implies $p_z(\phi)=1$ for all z, which is independent of m, hence the prior evidence exists for $m\to\infty$. This is nice, but hardly surprising.

The prior effective model dimension $N_{\vec{q}_*}$ is more interesting. $D=\phi$ implies $D_z=\phi$ implies $n_z=0$ implies $w(n_{z0},n_{z1})=1$ implies a 50/50 prior chance $g_z(\phi)=\frac{1}{2}$ for a split (see (12)). Recursion (11) reads

$$P_z[N_{\vec{q}_{z*}}=k+1]=\tfrac{1}{2}\sum_{i=0}^{k}P_{z0}[N_{\vec{q}_{z0*}}=i]\cdot P_{z1}[N_{\vec{q}_{z1*}}=k-i]$$

with $P_z[N_{\vec{q}_{z*}}=k]=\delta_{k0}$ for $l=m$ and $P_z[N_{\vec{q}_{z*}}=0]=\frac{1}{2}$ for $l<m$. So the recursion terminates in recursion depth $\min\{k+1,m-l\}$. Hence $P_z[N_{\vec{q}_{z*}}=k+1]$ is the same for all $m>l+k$, which implies that the limit $m\to\infty$ exists. Furthermore, recursion and termination are independent of z, hence also $a_k:=P_z[N_{\vec{q}_{z*}}=k]$. So we have to solve the recursion

$$a_{k+1}=\tfrac{1}{2}\sum_{i=0}^{k}a_i\cdot a_{k-i}\quad\text{with}\quad a_0=\tfrac{1}{2}\qquad(13)$$

The first few coefficients can be bootstrapped by hand: $(\frac{1}{2},\frac{1}{8},\frac{1}{16},\frac{5}{128},\frac{7}{256},\frac{21}{1024},\frac{33}{2048},...)$. A closed form can also be obtained: Inserting (13) into $f(x):=\sum_{k=0}^{\infty}a_kx^{k+1}$ we get $f(x)=\frac{1}{2}[x+f^2(x)]$ with solution $f(x)=1-\sqrt{1-x}$, which has Taylor expansion coefficients

$$a_k=(-)^k\binom{1/2}{k+1}=\frac{1}{2(k+1)4^k}\binom{2k}{k}\sim\frac{1}{2\sqrt{\pi}}\,k^{-3/2}$$

$(a_k)_{k\in\mathbb{N}_0}$ is a well-behaved distribution. It decreases fast enough to be a proper measure ($\sum_k a_k=f(1)=1<\infty$), but too slow for the expectation $E[N_{\vec{q}_*}]=\sum_k k\cdot a_k=\infty$ to exist. This is exactly how a proper non-informative prior on $\mathbb{N}$ should look like: as uniform as possible, i.e. slowly decreasing. Further, $P[N_{\vec{q}_*}<\infty]=\sum_k a_k=1$ implies $P[N_{\vec{q}_*}<\infty|D]=1$, which shows that the effective dimension is almost surely finite, i.e. infinite (Polya) trees have probability zero.

For the tree height we have $E_z[h_{\vec{q}_{z*}}(x)]=0$ if $l=m$ and otherwise

$$\begin{aligned}E_z[h_{\vec{q}_{z*}}(x)] &= \tfrac{1}{2}[1+E_{zx_{l+1}}[h_{\vec{q}_{zx_{l+1}*}}(x)]]\\ &= ... =1-(\tfrac{1}{2})^{m-l}\to 1\quad\text{for}\quad m\to\infty\end{aligned}$$

This also implies that the expected average height $E_z[\bar{h}_{\vec{q}_{z*}}]=1-(\frac{1}{2})^{m-l}\to 1$. This is the first case where the result is not independent of m for large finite m, but it converges for $m\to\infty$, what is enough for our purpose.

Single data item $D=(x)$. Since $p(x)\equiv 1$ (by symmetry and normalization) and $w_1=1$ are the same as for the $n=0$ case, all prior $n=0$, $m\to\infty$ results remain valid for $n=1$: $g(x)=\frac{1}{2}$, $P[N_{\vec{q}_*}=k|x]=a_k$, and $E[h_{\vec{q}_*}(x)|x]\to 1$.

General D. We now consider general D. For continuous spaces Γ and non-singular distribution $\dot{q}$, the probability of observing the same point more than once (multi-points) is zero and hence can, to a certain extend, be ignored. See [Hut04a] for a thorough workout of this case. In order to compute $p(D)$ and other quantities, we recurse (9) down the tree until D_z is either empty or a singleton $D_z=(x)\in\Gamma_z$. We call the depth $m_x:=\ell(z)$ at which this happens, the separation level. In this way, the recursion always terminates. For instance, for $\Gamma=[0,1)$, if $\varepsilon:=\min\{|x^i-x^j|:x^i\neq x^j$ with $x^i,x^j\in D\}$ is the shortest distance, then $m_x<\log_2\frac{2}{\varepsilon}=:m_0<\infty$, since $\varepsilon>0$. At the separation level we can insert the derived formulas for evidence, posterior, dimension, and height. Note, there is no approximation here. The procedure is exact, since we analytically computed the infinite recursion for empty and singleton D.

So we have devised a finite procedure, linear in the data size n, for exactly computing all quantities of interest in the infinite Bayes tree. In the worst case, we have to recurse down to level m_0 for each data point, hence our procedure has computational complexity $O(n\cdot m_0)$. For non-singular prior, the time is actually $O(n)$ with probability 1. So, inference in our mixture tree model is *very* fast. Posterior (weak) convergence/consistency for $m=\infty$ can be shown similarly to the $m<\infty$ case [Hut04a].

5 THE ALGORITHM

What it computes. In the last two sections we derived all necessary formulas for making inferences with our tree model. Collecting pieces together we get the exact algorithm for infinite tree mixtures below. It computes the evidence $p(D)$, the expected tree height $E[h_{\vec{q}_*}(x)|D]$ at x, the average expected tree height $E[\bar{h}_{\vec{q}_*}|D]$, and the model dimension distribution $P[N_{\vec{q}_*}|D]$. It also returns the number of recursive function calls, i.e. the size of the explicitly generated tree. The size is proportional to n for regular distributions $\dot{q}$.

The BayesTree algorithm (in pseudo C code) takes arguments $(D[],n,x,N)$; data array $D[0..n-1]\in[0,1)^n$, a point $x\in\mathbb{R}$, and an integer N. It returns $(p,h,\bar{h},\tilde{p}[],r)$; the logarithmic data evidence $p\widehat{=}\ln p(D)$, the expected tree height $h\widehat{=}E[h_{\vec{q}_*}(x)|D]$ at x, the average expected tree height $\bar{h}\widehat{=}E[\bar{h}_{\vec{q}_*}|D]$, the model dimension distribution $\tilde{p}[0..N-1]\widehat{=}P[N_{\vec{q}_*}=..|D]$, and the number of recursive function calls r i.e. the size of the generated tree. Computation time is about $N^2 n\log n$ nano-seconds on a 1GHz P4 laptop.

BayesTree($D[],n,x,N$)

⌈ if ($n\leq 1$ and ($n==0$ or $D[0]==x$ or $x\notin[0,1)$))
  ⌈ if ($x\in[0,1)$) then $h=1$; else $h=0$;
    $\bar{h}=1$; $p=\ln(1)$; $r=1$;
  ⌊ for($k=0,..,N-1$) $\tilde{p}[k]=a_k$; /* see (13) */
  else
  ⌈ $n_0=n_1=0$;
    for($i=0,..,n-1$)
    ⌈ if ($D[i]<\frac{1}{2}$) then[$D_0[n_0]=2D[i]$; $n_0=n_0+1$;]
    ⌊ else [$D_1[n_1]=2D[i]-1$; $n_1=n_1+1$;]
    $(p_0,h_0,\bar{h}_0,\tilde{p}_0[],r_0)$=BayesTree($D_0[],n_0,2x,N-1$);
    $(p_1,h_1,\bar{h}_1,\tilde{p}_1[],r_1)$=BayesTree($D_1[],n_1,2x–1,N–1$);
    $t=p_0+p_1-\ln w(n_0,n_1)$;
    if ($t<100$) then $p=\ln(\frac{1}{2}(1+\exp(t))$;
      else $p=t-\ln(2)$;
    $g=1-\frac{1}{2}\exp(-p)$;
    if ($x\in[0,1)$) then $h=g\cdot(1+h_0+h_1)$; else $h=0$;
    $\bar{h}=g\cdot(1+\frac{n_0+1}{n+2}\bar{h}_0+\frac{n_1+1}{n+2}\bar{h}_1)$;
    $\tilde{p}[0]=1-g$;
    for($k=0,..,N-1$) $\tilde{p}[k+1]=g\cdot\sum_{i=0}^{k}\tilde{p}_0[i]\cdot\tilde{p}_1[k-i]$;
  ⌊ $r=1+r_0+r_1$;
⌊ **return ($p,h,\bar{h},\tilde{p}[],r$);**

How algorithm BayesTree() works. Since evidence $p(D)$ and weight $1/w_n$ can grow exponentially with n, we have to store and use their logarithms. So the algorithm returns $p\widehat{=}\ln p(D)$. In the $n\leq 1$ branch, the closed form solutions $p\widehat{=}\ln p(\phi)=\ln(1)$, $h\widehat{=}E[h_{\vec{q}_*}(x)|\phi$ or $x]=1$, $\bar{h}\widehat{=}E[\bar{h}_{\vec{q}_*}|D]=1$, and $\tilde{p}[k]=a_k$ have been used to truncate the recursion. If $D=(x^1)\neq x$, we have to recurse further until x falls in an empty interval. In this case or if $n>1$ we partition D into points left and right of $\frac{1}{2}$. Then we rescale the points to [0,1) and store them in D_0 and D_1, respectively. Array D could have been reused (like in quick sort) without allocating two new arrays. Then, algorithm BayesTree() is recursively called for each partition. The results are combined according to the recursions derived in Section 2. $\ln w$ can be computed from (9) via $\ln n!=\sum_{k=1}^{n}\ln k$. (Practically, pre-tabulating a_k or $n!$ does not improve overall performance). For computing p we need to use $\ln(\frac{1}{2}(1+e^t))\doteq t-\ln 2$ to machine precision for large t in order to avoid numerical overflow.

Remarks. Strictly speaking, the algorithm has runtime $O(n\log n)$, since the sorting effectively runs once through all data at each level. If we assume that the data are presorted or the counts n_z are given, then the algorithm is $O(n)$ [Hut04a]. The complete C code, available from [Hut04a], also handles multi-points.

Note that x passed to BayesTree() is *not* and cannot be used to compute $p(x|D)$. For this, one has to call BayesTree() twice, with D and (D,x), respectively. The quadratic order in N is due to the convolution, which could be reduced to $O(N\log N)$ by transforming it to a scalar product in Fourier space with FFT.

Multiply calling BayesTree(), e.g. for computing the predictive density function $p(x|D)$ on a fine x-grid, is inefficient. But it is easy to see that if we once pre-compute the evidence $p_z(D_z)$ for all z up to the separation level in time $O(n)$, we can compute "local" quantities like $p(x|D)$ at x in time $O(\log n)$. This is because only the branch containing x needs to be recursed, the other branch is immediately available, since it involves the already pre-computed evidence only. The predictive density $p(x|D)=E[q(x)|D]$ and higher moments, the distribution function $P[x\leq a|D]$, updating D by adding or removing one data item, and most other local quantities can be computed in time $O(\log n)$ by such a linear recursion.

A good way of checking correctness of the implementation *and* of the derived formulas, is to force some *minimal* recursion depth m'. The results must be independent of m', since the closed-form speedups are exact and applicable anywhere beyond the separation level.

Numerical example. To get further insight into the behavior of our model, we numerically investigated some example distributions $\dot{q}()$. We have chosen elementary functions, which can be regarded as prototypes for more realistic functions. They include the Beta, linear, a singular, piecewise constant distributions with finite and infinite Bayes trees, and others. These examples on [0,1) also shed light on the other spaces discussed in Section 2, since they are isomorphic. The posteriors, model dimensions, and tree heights, of the singular distribution $\dot{q}(x)=2/\sqrt{1-x}$ are plotted in Figure 1 for random samples D of sizes $n=10^0,...,10^5$. The posterior $p(x|D)$ clearly converges for $n\to\infty$ to the true distribution $\dot{q}()$, accompanied by a (necessary) moderate growth of the effective dimension. For $n=10$ we show the data points. It is visible how each data point pulls the posterior up, as it should be ("one sample seldom comes alone"). The expected tree height $E[h(x)|D]$ correctly reflects the local needs for (non)splits, i.e. is larger near the singularity at $x=1$. The other examples display a similar behavior (see [Hut04a]).

6 DISCUSSION

We presented a Bayesian model on infinite trees, where we split a node into two subtrees with prior probability $\frac{1}{2}$, and uniform choice of the probability assigned to each subtree. We devised closed form expressions for various inferential quantities of interest at the data

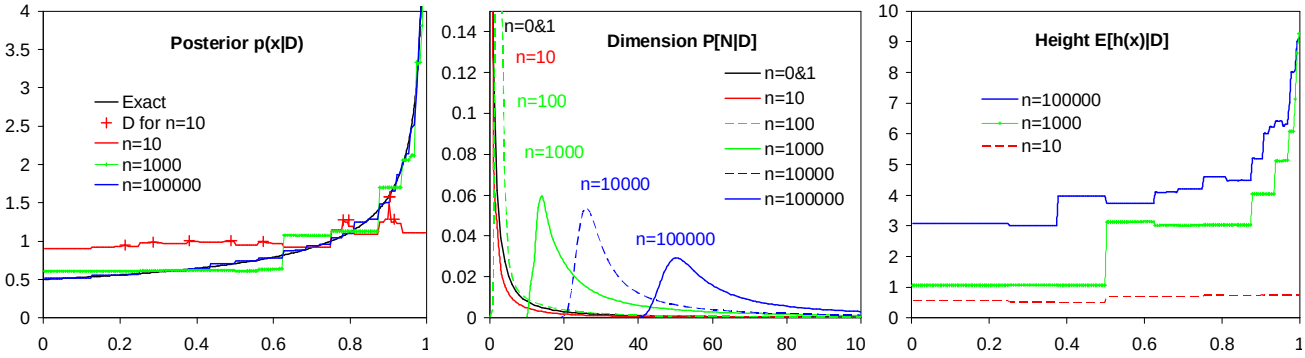


Figure 1: BayesTree() results for a prototypical proper singular distribution $\dot{q}(x)=2/\sqrt{1-x}$.

separation level, which led to an exact algorithm with runtime essentially linear in the data size. The theoretical and numerical model behavior was very reasonable, e.g. consistency (no underfitting) and low finite effective dimension (no overfitting).

There are various natural generalizations of our model. The splitting probability $p(s)$ could be chosen different from $\frac{1}{2}$, k-ary trees could be allowed, and the uniform prior over subtrees could be generalized to Beta/Dirichlet distributions. We were primarily interested in the case of zero prior knowledge, hence zero model (hyper)parameters, but the generalizations above make the model flexible enough, in case prior knowledge needs to be incorporated. The dependency on $p(s)$ is particularly interesting [Hut04a]. The expected entropy can also be computed by allowing fractional counts n_z and noting that $x\ln x = \frac{d}{dx}x^\alpha|_{\alpha=1}$ [Hut02]. A sort of maximum a posteriori (MAP) tree skeleton can also easily be read off from (9). A node Γ_z in the MAP-like tree is a leaf iff $\frac{p_{z0}(D_{z0})p_{z1}(D_{z1})}{w(n_{z0},n_{z1})}<1$. A challenge is to generalize the model from piecewise constant to piecewise linear continuous functions, at least for $\Gamma=[0,1)$. Independence of subtrees no longer holds, which was key in our analysis.

References

[DLR77] A. P. Dempster, N. Laird, and D. Rubin. Maximum likelihood estimation for incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, Series B 39:1–38, 1977.

[EW95] M. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90:577–588, 1995.

[Fer73] T. S. Ferguson. On the mathematical foundations of theoretical statistics. *Annals of Statistics*, 1(2):209–230, 1973.

[GM03] A. G. Gray and A. W. Moore. Nonparametric density estimation: Toward computational tractability. In *SIAM International Conf. on Data Mining*, volume 3, 2003.

[Goo83] I. J. Good. Explicativity, corroboration, and the relative odds of hypotheses. In *Good thinking: The Foundations of Probability and its applications*. University of Minnesota Press, Minneapolis, MN, 1983.

[Hut02] M. Hutter. Distribution of mutual information. In *Advances in Neural Information Processing Systems 14*, pages 399–406, Cambridge, MA, 2002. MIT Press.

[Hut04a] M. Hutter. Additional material to article. *http://www.idsia.ch/~marcus /ai/bayestreex.htm*, 2004.

[Hut04b] M. Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin, 2004. 300 pages, http://www.idsia.ch/~marcus/ai/uaibook.htm.

[Jay03] E. T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, Cambridge, MA, 2003.

[Lav94] M. Lavine. More aspects of Polya tree distributions for statistical modelling. *Annals of Statistics*, 22:1161–1176, 1994.

[Lem03] J. C. Lemm. *Bayesian Field Theory and Approximate Symmetries*. Johns Hopkins University Press, 2003.

[Mac03] D. J. C. MacKay. *Information theory, inference and learning algorithms*. Cambridge University Press, Cambridge, MA, 2003.

[PH04] J. Poland and M. Hutter. Convergence of discrete MDL for sequential prediction. In *Proc. 17th Annual Conf. on Learning Theory (COLT-2004)*, volume 3120 of *LNAI*, pages 300–314, Banff, 2004. Springer, Berlin.

[PW02] S. Petrone and L. Wasserman. Consistency of Bernstein polynomial posteriors. *Journal of the Royal Statistical Society*, B 64:79–100, 2002.

Restricted concentration models – graphical Gaussian models with concentration parameters restricted to being equal

Søren Højsgaard
Biometry Research Unit
Danish Institute of Agricultural Sciences
Research Center Foulum, DK–8830 Tjele
Denmark

Steffen Lauritzen
Department of Statistics
University of Oxford
1 South Parks Road, Oxford OX1 3TG
United Kingdom

Abstract

In this paper we introduce restricted concentration models (RCMs) as a class of graphical models for the multivariate Gaussian distribution in which some elements of the concentration matrix are restricted to being identical is introduced. An estimation algorithm for RCMs, which is guaranteed to converge to the maximum likelihood estimate, is presented. Model selection is briefly discussed and a practical example is given.

1 Introduction

This paper introduces a class of graphical Gaussian models, Lauritzen (1996), (hereafter abbreviated GGMs) also known as covariance selection models, Dempster (1972), in which elements of the concentration matrix are restricted to being identical. Such models are denoted *restricted concentration models* and abbreviated RCMs. These models are linear in the inverse covariance matrix and can therefore be seen as instances of models discussed by Anderson (1970). Besag (1974) also studies instances of such models.

RCMs can be of relevance in a variety of different problems. An example could be gene expression data where the expression of many genes are measured. From a biological point of view it may be of interest to embody in the model that the conditional covariance between genes i and j should be the same as the conditional covariance between genes k and l. It may also be of interest (and in some cases a necessity) to impose such restrictions simply in order to reduce the dimensionality of the problem.

Models with equal conditional correlations can be constructed within RCMs but this requires restrictions on both the conditional covariances and the conditional variances. An interesting extension of RCMs would therefore be models with equal conditional correlations and no other restraints.

Finally we mention that the restrictions in RCMs can lead to some regression functions being constrained to equality as illustrated in Section 3.

2 Background and notation

The setting in GGMs is i.i.d. samples of a random vector $y = (y_1, \ldots, y_d)^\top$ following a $N_d(\mu, \Sigma)$ distribution. Let $K = \Sigma^{-1}$ denote the inverse covariance matrix, also known as the concentration matrix with elements $(k^{\alpha\beta})$. It is then well known, Lauritzen (1996), p. 130, that the partial correlation between y_1 and y_2 given all other variables is

$$\rho_{12|3\ldots d} = -k^{12}/\sqrt{k^{11}k^{22}} \tag{1}$$

Thus $k_{12} = 0$ if and only if y_1 and y_2 are independent given all other variables, and this is the traditional focus of graphical Gaussian modeling.

A GGM is often represented by an undirected graph $G = (\Gamma, E)$ where Γ is the set of nodes representing the d variables and E is the set of undirected edges representing the concentration parameters $k^{\alpha\beta}$ which are not restricted to being zero. For additional properties of GGMs we refer to Lauritzen (1996), Chapter 5. In the following we use Greek letters to refer to variables and Latin letters to refer to sets of variables.

3 The problem to be solved

The issue addressed in this paper is to estimate K when some entries $k^{\alpha\beta}$ are restricted to being equal. Such restrictions can be imposed both on the diagonal and the off–diagonal elements of K.

Example To illustrate possible implications of such restrictions, consider the model in Figure 1. The asterisks indicate the restrictions that $k^{13} = k^{14} = c_1$,

$k^{23} = k^{24} = c_2$ and $k^{33} = k^{44} = c_3$, i.e.

$$K = \begin{bmatrix} k^{11} & k^{12} & c_1 & c_1 \\ k^{12} & k^{22} & c_2 & c_2 \\ c_1 & c_2 & c_3 & 0 \\ c_1 & c_2 & 0 & c_3 \end{bmatrix}$$

If we let $a = \{1, 2\}$ and $b = \{3, 4\}$, then the regression parameters when regressing b on a are given as $-(K^{bb})^{-1}K^{ba}$. Thus the slope parameters for y_3 and y_4 become identical,

$$E(y_i|y_1, y_2) = a_i + (c_1/c_3)y_1 + (c_2/c_3)y_2 \text{ for } i = 3, 4,$$

meaning that the regression lines are parallel.

Another property of this model is that some partial correlations are restricted to being equal. For example it follows directly from (1) that

$$\begin{aligned} \rho_{31|24} &= \rho_{41|23} = -c_3/\sqrt{k^{11}c_1} \text{ and} \\ \rho_{32|14} &= \rho_{42|13} = -c_3/\sqrt{k^{11}c_2}. \end{aligned}$$

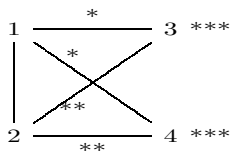


Figure 1: Graphical Gaussian model with the additional restrictions that (*) $k^{13} = k^{14} = c_1$, (**) $k^{23} = k^{24} = c_2$ and (***) $k^{33} = k^{44} = c_3$.

4 Restricted concentration models

To formalize the restrictions on the elements of K, let the edge set E be partitioned into non–empty disjoint subsets $E_1, \ldots, E_S$ each containing one or more edges. Let $E_1, \ldots, E_s$ denote those subsets containing more than one edge (those edges are said to be *marked*), and $E_{s+1}, \ldots, E_S$ be those containing only one edge (those edges are *unmarked*). Let $\Gamma_1, \ldots, \Gamma_T$ be a similar partitioning of Γ into sets of nodes, some of which are marked and some unmarked. A natural way to represent this situation graphically is to colour the edges (vertices) in the graph such that all edges (vertices) in the same edge (vertex) set have the same colour.

Let $\mathcal{R} = \{E_1, \ldots E_S, \Gamma_1, \ldots, \Gamma_T\}$ be the collection of such restrictions on K. The elements of $\mathcal{R}$ are in 1–1 correspondence with the parameters $\theta = (\theta_1, \ldots, \theta_{S+T})$ in $K = K(\theta)$. The *edge sets* $E_1, \ldots E_S$ and *vertex sets* $\Gamma_1, \ldots, \Gamma_T$ define a RCM.

5 The NIPS (Newton+IPS) algorithm

The algorithm is a combination of the classical IPS algorithm for graphical Gaussian models, Lauritzen (1996), p. 134 and the modified Newton procedure of Jensen, Johansen and Lauritzen (1991) (hereafter JJL91), see also Lauritzen (1996), p. 269.

Let $\hat{\Sigma} = \hat{K}^{-1}$ denote the current estimate of Σ at any time during the iteration, and let $f = n - 1$ where n is the number of observations, $SSD = \sum_{s=1}^{n}(y_s - \bar{y})(y_s - \bar{y})^\top$ and $S = SSD/f$.

5.1 Newton algorithm

The simplest version of the algorithm (which is just a specific version of the modified Newton algorithm of JJL91) is as follows:

Repeatedly loop through $\mathcal{R}$ until convergence doing the following: For each $s \in \mathcal{R}$ define the $d \times d$ matrix K^s as follows: 1) If s is an edge set, then K^s has entries $K^s_{\alpha\beta} = 1$ if $\{\alpha, \beta\} \in s$ and 0 otherwise. Thus K^s is the incidence matrix for the graph (Γ, s). 2) If s is a vertex set then K^s is a diagonal matrix with entries $K^s_{\alpha\alpha} = 1$ if $\alpha \in s$ and 0 otherwise. For convenience we shall identify a vertex α with a set $\{\alpha, \alpha\}$ such that vertex sets and edge sets can be treated simultaneously in the following.

Define the discrepancy $\Delta = tr(K^s\hat{\Sigma}) - tr(K^s S)$. For each element s do a sequence of Newton steps

$$\begin{aligned} \theta_s^{n+1} &\leftarrow \theta_s^n + \frac{\Delta}{tr(K^s\hat{\Sigma}K^s\hat{\Sigma}) + f\Delta^2/2}, \\ k^{\alpha\beta} &\leftarrow \theta_s^{n+1} \text{ for all } \{\alpha, \beta\} \in s. \end{aligned} \qquad (2)$$

The substitution (2) is repeated until convergence for the set s before moving on to the next set in $\mathcal{R}$. Thus the algorithm consists of two nested loops: 1) An outer loop running over the elements of $\mathcal{R}$ and 2) an inner loop maximizing L with respect to θ_s while keeping all other parameters fixed. Below it is shown that this algorithm in some cases can be speeded up by replacing the inner loop by a direct line search.

The likelihood equations are obtained as follows: With the definition of the matrices K^s for all $s \in \mathcal{R}$ given above, the concentration matrix can be written $K = \sum_s \theta_s K^s$. Let SS denote the sums–of–squares matrix. Then $tr(KSS) = \sum_s \theta_s tr(K^s SS)$. Let $t^s = \sum_s tr(K^s SS)$. Hence $(-t^1/2, \ldots, -t^{S+T}/2, \bar{y})$ is a set of canonical statistics, and these are to be equated with their expectation.

To do so, we exploit the following: The multivariate normal distribution is a regular k–dimensional exponential family. Therefore the maximum likelihood estimate (MLE) exists and is unique, provided that the

sufficient statistic is contained in its convex support. By Theorem 2 in Jensen *et al.* (1991), the MLE can be found by iteratively maximizing over each canonical parameter, keeping the others fixed. Note that when keeping all parameters but one at fixed values we get a regular one–dimensional exponential family. By Theorem 1 in JJL91, their modified Newton algorithm applied to a one–dimensional regular exponential family converges to the MLE for any starting value.

Following Lauritzen (1996), p. 133, $\hat{\mu} = \bar{y}$ so what remains is to maximize $L(\theta, \hat{\mu})$ over Θ which is an $S+T$ dimensional space restricted only by the requirement that $K(\theta)$ must be positive definite for all $\theta \in \Theta$. For any $\theta^* \in \Theta$ and any $s \in \mathcal{R}$, define

$$\Theta_s(\theta^*) = \{\theta \in \Theta | \theta_r = \theta_r^* \text{ for } r \neq s\}.$$

Then L is maximized by cyclically maximizing L over $\Theta_s(\theta^*)$, Lauritzen (1996), p. 270. For practical reasons we have chosen to fit the model on S rather than on SS. Following Lauritzen (1996) p. 259, $\tau^s = E(t^s) = -\frac{1}{2}tr(K^s\Sigma)$ and $v^s = Var(t^s) = \frac{1}{2}tr(K^s\Sigma K^s\Sigma)$. The modified Newton algorithm of JJL91 consists in updating θ as

$$\theta^{n+1} = \theta^n + \frac{t^s - \tau^s}{v^s + (t^s - \tau^s)^2}$$

which specializes to (2) in this context.

Convergence The parameter space Θ is restricted by $K(\theta)$ having to be positive definite. We have not shown that the Newton steps are guaranteed to keep K positive definite and this should therefore strictly speaking be checked at each Newton step, decreasing the step length appropriately if the condition is no longer satisfied, see JJL91. Empirical evidence suggests however, that K indeed remains positive definite.

5.2 IPS algorithm

For a GGM (without restrictions of the kind discussed in this paper) let $a = \{\alpha, \beta\}$ be an edge in the graph and let b denote the complement to a. Then in the IPS algorithm, see e.g. Lauritzen (1996) p. 134 ff, can be used for updating the parameters $k^{\alpha\alpha}, k^{\beta\beta}$ and $k^{\alpha\beta}$ by updating the 2×2 submatrix K^{aa} of K as

$$K^{aa} \leftarrow (S^{aa})^{-1} + K^{ab}(K^{bb})^{-1}K^{ba}. \qquad (3)$$

Note that in this step both the conditional variances and conditional covariances are updated. This IPS step maximizes the likelihood over the particular section of the parameter space given by $k^{\alpha\alpha}, k^{\beta\beta}$ and $k^{\alpha\beta}$ and thus no iteration is needed. This operation can also be performed on a single vertex α, which gives an update of the 1×1 submatrix $K^{\alpha\alpha}$.

5.3 NIPS algorithm

Considerable computational savings can be achieved by combining the Newton sequence (2) with the IPS step (3) and this combination constitutes the NIPS (=Newton+IPS) algorithm. The combination is straight forward and most easily explained by an example: The graph in Figure 2 has cliques [12][23][34][45]. The asterisks indicate that the edges [12] and [23] and the vertices 2 and 3 are *marked*, i.e. the restrictions $k^{12} = k^{23}$ and $k^{22} = k^{33}$.

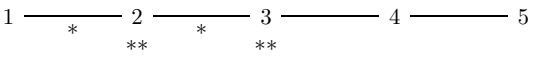


Figure 2: RCM with the additional restrictions that (*) $k^{12} = k^{23}$ and (**) $k^{22} = k^{33}$.

The marked entries can be updated using the Newton sequence while k^{11} is unrestricted and can be updated using an IPS step on a 1×1 matrix. The edge [45] (comprising the parameters k^{44}, k^{45} and k^{55} can also be updated in a single IPS step on a 2×2 matrix. Left to consider is therefore only k^{34}. Even though no restriction is put onto this parameter it can not immediately be updated using an IPS step (3) because that would also update k^{33} (and k^{44}) which is constrained. Therefore this parameter is updated using a Newton sequence. This constitutes one full cycle of the inner loop of the NIPS algorithm. Note that it is easy to keep track of such restrictions: Whenever an edge $\{\alpha, \beta\}$ contains a marked vertex, the edge must itself be marked.

Computational Savings The following considerations can lead to substantial computational savings:

1. Computational savings can be achieved when calculating $\Delta = tr(K^s\hat{\Sigma}) - tr(K^sS)$. The incidence matrix K^s serves to pick out (and sum the correct way) the relevant entries of S and $\hat{\Sigma}$. For a fixed edge set $s \in \mathcal{R}$, let a denote the set of vertices in s and let b be the complement of a. Let $\tilde{A}^s$ be the incidence matrix for the graph (a, s) and let finally $\hat{\Sigma}^{aa}$ and S^{aa} denote the corresponding submatrices of $\hat{\Sigma}$ and S. It it then straight forward to see that $tr(K^sS) = tr(\tilde{A}^sS^{aa})$ and hence $\Delta = tr(\tilde{A}^s\hat{\Sigma}^{aa}) - tr(\tilde{A}^sS^{aa})$. The modification when s is a vertex set is straight forward.

2. After updating entries of K in a NR step, one need not find $\Sigma = K^{-1}$. The relevant part Σ^{aa} can be found as $(K^{aa} - K^{ab}(K^{bb})^{-1}K^{ba})^{-1}$, and here it is noted that 1) $K^{ab}(K^{bb})^{-1}K^{ba}$ is fixed throughout the whole Newton sequence and 2) the dimension of Σ^{aa} is often much smaller than the dimension

of Σ. A similar construct can be used when calculating the value of the likelihood function.

3. Convergence is sometimes speeded up when replacing the Newton steps in (2) by an alternative line search algorithm of the form

$$\begin{aligned} \theta_s^{n+1} &\leftarrow \theta_s^n + \alpha \cdot p, \\ k^{\alpha\beta} &\leftarrow \theta_s^{n+1} \text{ for all } \{\alpha, \beta\} \in s \end{aligned} \qquad (4)$$

where $p = \frac{\Delta}{tr(A^s \hat{\Sigma} A^s \hat{\Sigma}) + f\Delta^2/2}$ and α is chosen to maximize L in the direction defined by $\theta_s^n + tp$.

4. If a clique consists exclusively of unmarked edges/vertices, then it is more computationally efficient to update the entire clique using IPS at one time rather than working the way through the edges one at the time.

6 Implementation

The algorithm has been implemented in the general statistical package `R`, R Development Core Team (2004).

7 Model selection issues

The number of different models which can be formed by colouring edges/vertices in a given graph is enormous. To illustrate the complexity, consider graphs with vertices, 1,2 and 3 (for which there are 8 different graphs). There are 5 possible vertex sets: $\{123\}, \{12, 3\}, \{1, 23\}, \{13, 2\}$ and $\{1, 2, 3\}$. A tedious calculation shows that there are in total (over all 8 graphs) 15 possible vertex sets giving $5 \times 15 = 75$ different models! Therefore, good model selection strategies become important. Here we shall just outline some ideas:

Often in model selection in graphical models one consider the operations `dropEdge` and `addEdge`. For RCMs there are four additional operations which are natural to consider: `joinEdgeSet` and `splitEdgeSet` (and similarly for vertices). In connection with a backward model search where edges are successively deleted it is tempting to supplement with the possibility of joining two edge sets. If there are p edge sets then there are $p(p-1)/2$ pairwise comparisons of the corresponding parameters and this can be done by e.g. calculating Wald statistics (which requires $Var(\hat{\theta})$ to be computed).

A more brute force approach is to search for a graphical model and then apply a clustering algorithm to the diagonal of K and to the non–zero off–diagonal elements of K.

One motivation for considering RCMs is applications where data is sparse, i.e. where $n < d$. In this case S is singular and hence $K = S^{-1}$ does not exist. One option in this case is to start from the independence model and do a forward selection possibly supplied with joining operations as discussed above.

8 Example: measurements on pig carcasses

To illustrate the developments in this paper we consider a prediction problem: In slaughter pig production, prediction of the lean meat content is important 1) to ensure fair payment to the producers and 2) to ensure an appropriate processing of the meat afterwards. The task is to predict the lean meat percentage y on the basis of a set of predictor variables denoted by x. In modern carcass grading, the predictor variables are often obtained e.g. by ultra sound measurements on the carcass and hence the number of predictor variables can be very large – and much larger than the sample size.

For simplicity, we here consider the `carcass` data set contained in the `mimR` package in `R`, see Højsgaard (2004). This data set contains measurements of the thickness of the meat and fat layer at three locations on the back of 340 carcasses. The data also contains the lean meat percentage determined by dissection. The response variable is the meat percentage, $y = MP$ while x denotes the measurements of thickness of meat and fat layers. The regression coefficients for the prediction are $\Sigma_{yx}\Sigma_{xx}^{-1} = -(K^{yy})^{-1}K^{yx}$. The problem in such prediction problems is that either Σ_{xx} is singular or it is very ill–conditioned because the predictor variables often are very correlated.

To accommodate for this, one often make a principal component regression or a partial least squares regression to obtain the regression coefficients. Other alternatives are ridge regression and the lasso, see e.g. Hastie, Tibshirani and Friedman (2001), pp. 59 for a description of these methods.

8.1 Selection of different models

The saturated model (which has Table 1 as concentration matrix) is in the following denoted $\mathcal{M}1$. Table 1 shows that the fat–concentration parameters all tend to be of the same size (conditional variances as well as covariances) and so do the meat concentration parameters. Similarly, the concentration parameters between the fat measurements and the lean meat percentage appear identical and so do (to a lesser extent) the concentration parameters between the meat measurements and the lean meat percentage. The model

with these constraints is denoted $\mathcal{M}1r$ and the estimated concentration matrix is shown in Table 2.

Table 1: Empirical concentration matrix for the carcass data (multiplied by 10).

	F1	F2	F3	M1	M2	M3	MP
F1	4.36	-1.99	-1.58	0.28	-0.73	0.41	0.99
F2	-1.99	5.35	-2.09	-0.26	0.64	-0.53	0.88
F3	-1.58	-2.09	5.57	-0.56	-0.06	0.26	0.71
M1	0.28	-0.26	-0.56	1.58	-0.60	-0.56	-0.33
M2	-0.73	0.64	-0.06	-0.60	1.35	-0.88	-0.04
M3	0.41	-0.53	0.26	-0.56	-0.88	1.57	-0.14
MP	0.99	0.88	0.71	-0.33	-0.04	-0.14	2.63

Table 2: Estimated concentration matrix for the carcass data (multiplied by 10) under the model $\mathcal{M}1r$ with parameters restricted to being equal.

	F1	F2	F3	M1	M2	M3	MP
F1	4.83	-1.77	-1.77	0.30	-0.86	0.42	0.88
F2	-1.77	4.83	-1.77	-0.27	0.58	-0.37	0.88
F3	-1.77	-1.77	4.83	-0.30	-0.04	0.04	0.88
M1	0.30	-0.27	-0.30	1.40	-0.64	-0.64	-0.16
M2	-0.86	0.58	-0.04	-0.64	1.40	-0.64	-0.16
M3	0.42	-0.37	0.04	-0.64	-0.64	1.40	-0.16
MP	0.88	0.88	0.88	-0.16	-0.16	-0.16	2.64

Starting with the independence model and doing a forward selection we get the model $\mathcal{M}2$ with concentration matrix in Table 3. Then we applied a clustering algorithm to the diagonal and to the off–diagonals to identify possible edge sets and vertex sets. Inspired by Table 1, we asked for 3 clusters on the diagonal and 5 clusters on the off–diagonal. The model with these restrictions is $\mathcal{M}2r$ and the estimated concentrations are presented in Table 4.

This scheme was repeated with a backward selection starting from the saturated model giving model $\mathcal{M}3$. Clustering the entries as described above gave $\mathcal{M}3r$. (The estimated concentration matrices have been omitted).

Table 3: Estimated concentration matrix for the carcass data (multiplied by 10) for $\mathcal{M}2$.

	F1	F2	F3	M1	M2	M3	MP
F1	4.06	-1.68	-1.53	0.00	-0.18	0.00	1.08
F2	-1.68	5.04	-2.12	0.00	0.00	0.00	0.78
F3	-1.53	-2.12	5.54	-0.39	0.00	0.00	0.75
M1	0.00	0.00	-0.39	1.52	-0.56	-0.56	-0.27
M2	-0.18	0.00	0.00	-0.56	1.22	-0.79	0.00
M3	0.00	0.00	0.00	-0.56	-0.79	1.51	-0.26
MP	1.08	0.78	0.75	-0.27	0.00	-0.26	2.68

8.2 Model comparisons – predictive performance

To evaluate the feasibility of the various models, we took a cross validation approach as follows: Out of the 340 carcasses we took a random sample of size $N = 8, 10, 15, 20, 30$ and fitted the models to these training data. Then we predicted MP for the validation data consisting of $340–N$ carcasses and calculated the mean squared prediction error (MSPE) defined as $\frac{1}{340-N}\sum_i (y_i - \hat{y}_i)^2$. This scheme was repeated $M = 5$ times and at the end average MSPE was calculated. To provide a benchmark for comparison we also made a principal component regression (PCR) and a partial least squares regression (PLS). Højsgaard, Jørgensen, Olsen and Busk (2004) have found that 3 components were optimal in PLS and PCR for predictions of these data, and therefore 3 components have been used here. To ease the comparison the MSPEs were all calculated relative to the MSPE for the PCR model.

Table 4: Estimated concentration matrix for the carcass data (multiplied by 10) for $\mathcal{M}2r$.

	F1	F2	F3	M1	M2	M3	MP
F1	4.60	-2.00	-2.00	0.00	-0.20	0.00	0.77
F2	-2.00	4.60	-1.11	0.00	0.00	0.00	0.77
F3	-2.00	-1.11	4.60	-0.20	0.00	0.00	0.77
M1	0.00	0.00	-0.20	1.06	-0.47	-0.47	-0.20
M2	-0.20	0.00	0.00	-0.47	1.06	-0.47	0.00
M3	0.00	0.00	0.00	-0.47	-0.47	1.06	-0.20
MP	0.77	0.77	0.77	-0.20	0.00	-0.20	2.39

8.3 Results

The relative MSPEs are presented in Table 5. Within each sample size, we find the following: It is always beneficial to reduce the saturated model $\mathcal{M}1$ to the restricted model $\mathcal{M}1r$, and for small samples ($N = 8, 10$) the improvement is quite dramatic. (Note that when $N = 8$ the saturated model is just identifiable as there are 7 variables in the model).

A comparison of models $\mathcal{M}i$ and $\mathcal{M}ir$ for $i = 2, 3$ yields no clear picture, but it suggests that there is a place for refinement of the brute force clustering approach used in getting from $\mathcal{M}i$ and $\mathcal{M}ir$. For each sample size, one of the RCMs always performs at least as well or better than the traditional regression methods PLS and PCR. Finally it is noted that when sample size increases the models perform more and more similarly, which was to be expected.

8.4 Computing time

Compared with the IPS algorithm used for GGMs the NIPS algorithm presented here is somewhat more time consuming. For example, fitting $\mathcal{M}3$ (which is a GGM) took 1.27 seconds while fitting the RCM $\mathcal{M}3r$ took 4.87 seconds.

Table 5: Relative mean squared prediction error (MSPE) (calculated relative to MSPE for principal component regression) for different models and different sizes of the training data sets.

	Sample size				
	8	10	15	20	30
$\mathcal{M}1$	4.53	1.08	1.10	1.06	1.01
$\mathcal{M}1r$	0.99	0.92	0.99	0.99	0.99
$\mathcal{M}2$	1.11	1.03	1.04	1.03	1.01
$\mathcal{M}2r$	1.18	0.99	1.03	0.99	1.00
$\mathcal{M}3$	1.20	1.04	1.04	1.07	1.00
$\mathcal{M}3r$	1.20	0.95	1.01	1.00	1.01
PLS	1.16	1.01	1.03	1.04	1.01
PCR	1.00	1.00	1.00	1.00	1.00

9 Discussion and directions for future work

This paper has presented an estimation algorithm for restricted concentration models (RCMs), and it has been proven empirically that important gains in terms of prediction precisions can be achieved from such models.

It is emphasized, that to use the result in JJL91 we should strictly speaking check that the concentration matrix stays positive definite in each step (2) and, if not, only move half of the distance to the associated boundary point of the parameter space. We have not seen an example where the positive definiteness has been violated, but we have not been able to prove theoretically that this cannot happen. For practical purposes we therefore suggest that this check is only performed occasionally.

To make RCMs of practical importance, it is important to investigate possible model selection strategies for RCMs, and this is a subject of future work. In this connection it will become important to make a fast implementation of the NIPS algorithm.

References

Anderson, T. W. (1970). Estimation of covariance matrices which are linear combinations or whose inverses are linear combinations of given matrices. In: *Essays in Probability and Statistics* (eds. R. C. Bose, I. M. Chakravarti, P. C. Mahalanobis, C. R. Rao and K. J. C. Smith), University of North Carolina Press, Chapel Hill, N.C., 1–24.

Besag, J. E. (1974). Spatial interaction and the statistical analysis of lattice systems (with discussion). *Journal of the Royal Statistical Society, Series B* **36**, 192–236.

Dempster, A. P. (1972). Covariance selection. *Biometrics* **28**, 157–175.

Hastie, T., Tibshirani, R. and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer.

Højsgaard, S. (2004). The mimR package for graphical modelling in R. *Journal of Statistical Software* .

Højsgaard, S., Jørgensen, E., Olsen, E. V. and Busk, H. (2004). A comparison of latent variable models and partial least squares regression – with an application to pig carcass grading. *Livestock Production Science* Manuscript submitted.

Jensen, S. T., Johansen, S. and Lauritzen, S. L. (1991). Globally convergent algorithm for maximizing likelihood function. *Biometrika* **78**, 867–877.

Lauritzen, S. L. (1996). *Graphical Models*. Oxford University Press.

R Development Core Team (2004). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-00-3.

Fast maximum *a posteriori* inference in Monte Carlo state spaces

Mike Klaas **Dustin Lang** **Nando de Freitas**

Computer Science Department
University of British Columbia
{klaas,dalang,nando}@cs.ubc.ca

Abstract

Many important algorithms for statistical inference can be expressed as a weighted max-kernel search problem. This is the case with the Viterbi algorithm for HMMs, message construction in maximum *a posteriori* BP (max-BP), as well as certain particle-smoothing algorithms. Previous work has focused on reducing the cost of this procedure in discrete regular grids [4]. Monte-Carlo state spaces, which are vital for high-dimensional inference, cannot be handled by these techniques. We present a novel dual-tree based algorithm that is appliable to a wide range of kernels and shows substantial performance gains over naïve computation.

Introduction

Max-kernel problems arise at the heart of many powerful and widely-used statistical inference algorithms. Examples include the message computation in max belief propagation, sequence recursion in the Viterbi algorithm, and classes of maximum *a posteriori* sequence estimation algorithms based on particle methods. This operation is expensive—requiring $O(N^2)$ operations, where N is the size of the state space of a random variable. As a result, applications with large state spaces either must artificially coarsen the state space or simply choose to use less powerful inference techniques. Recent work by Felzenszwalb *et al.* addresses the computational burden when the state space can be embedded in a regular discrete grid [4, 5]. This technique, based on the *distance transform*, is extremely powerful in its domain, but has two major limitations:

- It is limited to kernels of the form $K(x, y) = \exp\left\{\frac{1}{\sigma^2}\|x - y\|\right\}$ or $\exp\left\{\frac{1}{\sigma^2}\|x - y\|^2\right\}$
- It is only applicable to state spaces embedded in a regular grid of parameters.

Monte Carlo methods, such as MCMC and particle filters, have been shown to effectively adapt to examine interesting regions of the state space, and can achieve better results than regular discretizations using fewer support points [1, 13]. Problems requiring high-dimensional inference are ubiquitous in machine learning, and are best attacked with Monte Carlo techniques as regular discretizations grow exponentially and quickly become intractable. In this paper, we address the need of fast algorithms for computing weighted max-kernel on Monte Carlo grids by demonstrating how the quadratic cost can be reduced to $N \log N$ by adopting and extending powerful algorithms proposed for N-body simulation [7, 8].[1]

In particular, we develop a new efficient dual-tree recursion to exactly solve the max-kernel problem. We derive the method in the context of kernels parameterized by a distance function,[2] which represent a broad class of frequently used kernel functions, including Gaussians, Epanechnikov, spherical, and linear kernels, as well as thresholded versions of the same. Our method can also be used to accelerate other spatial-based kernels (such as $K(x, y) = x \cdot y$), and problems that have multiple kernels over different regions of the state space, but we restrict our attention to the simpler and more common case in this paper.

Our empirical results show that our algorithm provides a speedup of several orders of magnitude over the naïve method, becoming more efficient after as little as 10ms of compute time. The dual-tree algorithm still compares favorably to naïve computation on discrete grids where the distance transform can be applied, but we

[1] We note that there are techniques for dealing with KDE on Monte Carlo grids (fast Gauss Transform), but these are inapplicable in the max-kernel setting.

[2] By *distance functions* we mean functions that are similar to a metric but need not obey the triangle inequality.

find that the latter algorithm is superior in this case.

The performance of algorithms based on dual-tree recursion as N grows is relatively well-understood; see Gray and Moore [8] and Ihler [9]. However, we have found that the performance of this family of techniques also depends heavily on other variables, such as the data distribution, the dimensionality of the problem, and the choice of spatial index and kernel. We present several experiments to investigate these effects, and we believe that the conclusions can be generalized to other pruning-based dual-tree algorithms.

1 Problem setting

The algorithms we discuss in this paper are designed to solve the following problem: We are given points (which we will call *particles*) $X \triangleq \{x_j\}$ and $Y \triangleq \{y_i\}$, and weights $\{w_j\}$ corresponding to the X particles. The source (X) particles exert an *influence* on the target (Y) particles given by $\text{infl}(x_j, y_i) = w_j K(x_j, y_i)$, where $K(\cdot)$ is an affinity kernel. We wish to compute, for each y, the maximum influence attained and the x particle corresponding to it,[3] ie.

$$f_i = \max_{j=1}^{N} w_j K(y_i, x_j) \quad i = 1, 2, \ldots, M \tag{1}$$

This procedure's $O(MN)$ cost dominates the runtime of many important algorithms such as max-BP and MAP sequence estimation, which limits their use to settings of small order (correponding to a coarse discretization of a continuous state space or choosing a small number of particles).

In the following section, we detail how the max-kernel algorithm arises in common inference methods.

1.1 Maximum *a posteriori* belief propagation

Given a graphical model with latent variables $\mathbf{u}_{1:n}$[4], observations $\mathbf{z}_{1:n}$, and potentials ψ_{kl}, ϕ_k, a joint probability distribution is admitted:

$$p(\mathbf{u}_{1:n}, \mathbf{z}_{1:n}) = \frac{1}{Z} \prod_{k,l} \psi_{kl}(\mathbf{u}_k, \mathbf{u}_l) \prod_k \phi_k(\mathbf{u}_k, \mathbf{z}_k)$$

We are interested in computing the maximum *a posteriori* estimate, $\mathbf{u}_{1:n}^{MAP} = \arg\max_{\mathbf{u}_{1:n}} p(\mathbf{u}_{1:n}|\mathbf{z}_{1:n})$. We can use the standard max-product belief propagation equations for message passing and marginal (belief) computation [12]. The message from node l to node k is given by:

$$m_{lk}(u_{ki}) = \max_{j=1}^{|\mathbf{u}_l|} \phi(u_{lj}, \mathbf{z}_l)\psi(u_{ki}, u_{lj}) \prod_{r \in \mathcal{N}(l)-k} m_{rl}(u_{lj}) \tag{2}$$

where $\mathcal{N}(l)-k$ denotes the neighbours of node l excluding k. We can re-write equation (2) as a max-kernel problem by setting

$$\begin{aligned} \{x_j\} &= \{u_{lj}\}, \\ \{y_i\} &= \{u_{ki}\}, \\ w_j &= \phi(u_{lj}, \mathbf{z}_l) \prod_{r \in \mathcal{N}(l)-k} m_{rl}(u_{lj}), \\ K(y_i, x_j) &= \psi(u_{ki}, u_{lj}) \end{aligned}$$

1.2 MAP sequence estimation using particle methods

Consider the Markovian time-series model with latent state $\mathbf{u}_t$ and observations $\mathbf{z}_t$ given by

$$\begin{aligned} \mathbf{u}_t &\sim p(\mathbf{u}_t|\mathbf{u}_{t-1}) \\ \mathbf{z}_t &\sim p(\mathbf{z}_t|\mathbf{u}_t) \end{aligned}$$

In standard particle filtering [3], we draw a set of samples $\left\{u_{1:n}^{(i)}\right\}_{i=1}^{N}$ using sequential importance sampling in order to approximate the filtering distribution with a Monte Carlo estimator $\widehat{p}(\mathbf{u}_n|\mathbf{z}_{1:n}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{u_n^{(i)}}(d\mathbf{u}_n)$, where $\delta_{u_n^{(i)}}(d\mathbf{u}_n)$ denotes the delta Dirac function. This is typically done in a chain (or tree) with n nodes, at a cost of $O(nN)$. However, our goal is to obtain an estimate of the maximum *a posteriori* sequence

$$\mathbf{u}_{1:n}^{MAP}(n) \triangleq \arg\max_{\mathbf{u}_{1:n}} p(\mathbf{u}_{1:n}|\mathbf{z}_{1:n}). \tag{3}$$

As introduced by Godsill *et al.* in [6], equation (3) can be estimated by performing a Viterbi-like algorithm on the Monte Carlo state space induced by the filtered particles at time t. At the heart of this algorithm lies the following recursion:

$$\begin{aligned} \delta_k(j) = &\log p\left(\mathbf{z}_k|u_k^{(j)}\right) \\ &+ \max_i \left[\delta_{k-1}(i) + \log p\left(u_k^{(j)}|u_k^{(i)}\right)\right] \end{aligned} \tag{4}$$

This must be computed for each particle, thus incurring a $O(N^2)$ cost. It is straightforward to show that the maximization in (4) is equivalent to the max-kernel problem in equation (1) transformed to log space.

[3] We will subsequently refer to this procedure as *weighted maximum-kernel*, or simply *max-kernel*.

[4] In describing these algorithms, we use $\mathbf{u}$ and $\mathbf{z}$ rather than the traditional x and y to highlight the distinction between the variables in the inference algorithms and the variables in the max-kernel computation.

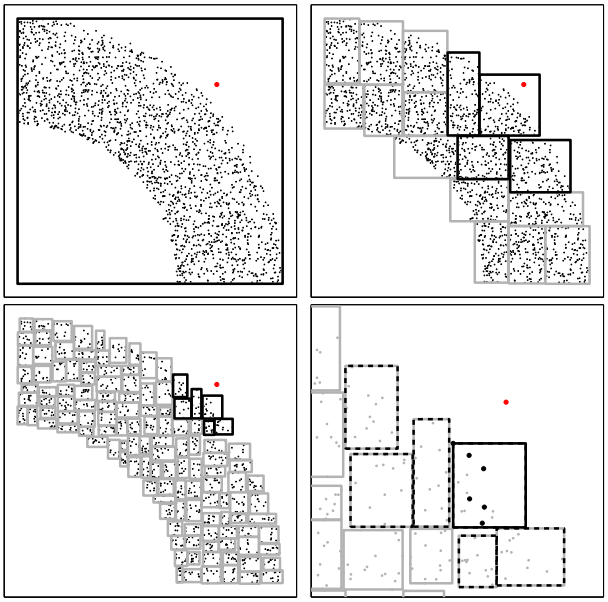


Figure 1: Example of pruning the max-kernel algorithm (for a single y particle). The candidate (dark) and non-candidate (light) nodes are shown. In the bottom-right plot, a close-up of the six final candidate nodes is shown (dashed). The single box whose particles are examined is shown in black. The subset of the particles that are examined individually is shown in black. There were 2000 particles in X, of which six nodes (containing 94 particles total) were candidate leaf nodes. Of these, only six particles from the first node were examined individually.

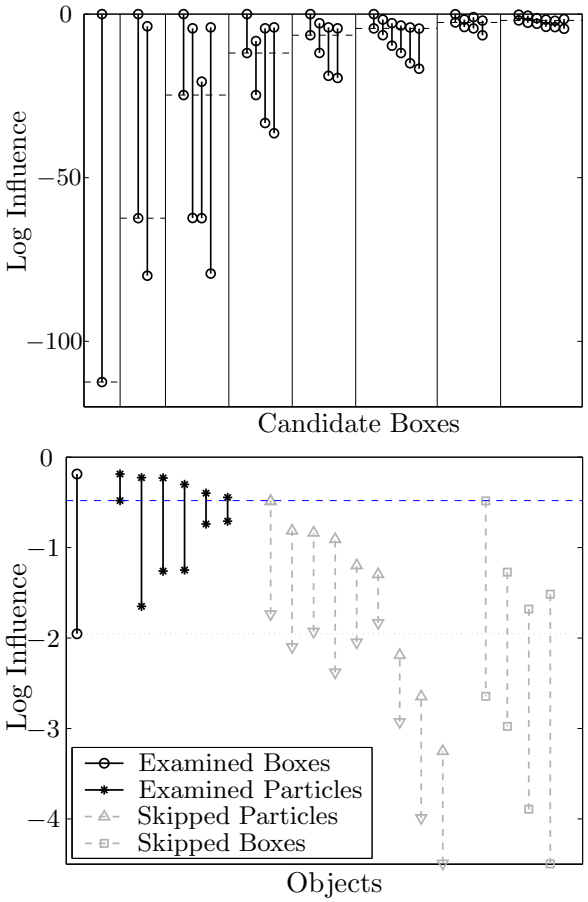


Figure 2: Dual-tree max-kernel example. *Top:* the influence bounds for the nodes shown in figure 1. The pruning threshold at each level is shown (dashed line), along with the bounds for each candidate node. *Bottom:* pruning at the leaf level: in the example, six leaf nodes are candidates. We begin examining particles in the first box. As it happens, the first particle we examine is the best particle (the correct answer). Pruning by particle weight (the upper marker) allows us to ignore all but the first six particles. The pruning threshold is then sufficiently high that we can prune the remaining candidate nodes without having to examine any of their particles.

2 Fast methods for computing max-kernel

2.1 The distance transform

In [4], Felzenszwalb and Huttenlocher derive a fast algorithm for a class of max-kernel problems by observing that the maximization in equation (1) is solvable by applying the distance transform. This achieves $O(N)$ cost and is very efficient in practice. Additionally, the problem is separable in dimensionality, so a d-dimensional transform of N^d points costs $O(dN^d)$.

2.1.1 Extension to Monte Carlo grids in 1-D

While the distance transform was designed to work exclusively on regular grids, it is easily extended to irregular grids in the one-dimensional case, for a small increase in cost. This observation is novel, to our knowledge, although it represents a rather direct extension to the original algorithm.

Assume we are given source particles $\{x_1, \ldots, x_N\}$ and target particles $\{y_1, \ldots, y_M\}$. The first step of the algorithm is to compute the lower envelope of the parabolas anchored at $\{x_i\}$. This step is unchanged, save that the x particles need to be pre-sorted at a cost of $O(N \log N)$. The second step is to calculate the value of the lower envelope at each y particle. This can be done by either pre-sorting the y particles, or employing binary search on the lower-envelope, which costs $O(M \log M)$ or $O(M \log N)$ respectively.

Unfortunately, this extension only applies to the one-dimensional case. Other means must be used to compute higher-dimensional max-kernel problems on Monte Carlo grids.

```
INPUTS: root nodes of X and Y trees: X_r, Y_r.
ALGORITHM:
  leaves = {}, candidates = {X_r}
  max_recursive(Y_r, leaves, candidates, −∞)
FUNCTION  max_recursive(Y, leaves, candidates, τ)
  if (leaf(Y)  AND  candidates = {})
    // Base Case: reached leaves (see figure 4).
    max_base_case(Y, leaves)
  else // Recursive case: recurse on each Y child.
    foreach y ∈ children*(Y)
      τ_y = τ, valid = {}
      foreach p ∈ candidates
        // Check if we can prune parent node p.
        if (w(p) K(d^l(p, y)) < τ_y)
          continue
        foreach x ∈ children(p)
          // Compute child bounds.
          f^{u,l}(x) = w(x) K(d^{l,u}(x, y))
        // Set pruning threshold.
        τ_y = max(τ_y , max_x(f^l(x)))
        valid = valid ∪ {x ∈ children(p) : f^u(x) ≥ τ_y}
      valid = {x ∈ valid : f^u(x) ≥ τ_y}
      leaves_y = {x ∈ valid : leaf(x)}
      candidates_y = {x ∈ valid : NOT(leaf(x))}
      sort(leaves_y by f^l)
      max_recursive(y, leaves_y, candidates_y, τ_y)
```

Figure 3: Dual-tree max-kernel algorithm, part 1.

```
FUNCTION  max_base_case(Y, leaves)
  foreach x ∈ leaves
    f^{u,l}(x) = w(x) K(d^{l,u}(x, Y))
  τ = max_x(f^l(x))
  leaves = {x ∈ leaves : f^u(x) ≥ τ}
  sort(leaves by f^l)
  // Examine individual y points.
  foreach y ∈ Y
    τ_y = τ
    foreach x ∈ leaves
      // Prune nodes by Y (cached), then by y.
      if (f^u(x) < τ_y OR w(x) K(d^l(x, y)) < τ_y)
        continue
      // Examine individual x particles.
      foreach i ∈ x
        // Prune by weight.
        if (f^u(x) w(i)/w(x) < τ_y)
          break
        f(i) = w(i) K(d(i, y))
        if (f(i) > τ_y)
          // i is the new best particle.
          τ_y = f(i), x*(y) = i
```

Figure 4: Dual-tree max-kernel algorithm, part 2.

2.2 Dual-tree max-kernel

In this section we present a novel dual-tree algorithm for solving the weighted max-kernel problem. Our algorithm is based on bounding the distance and weight, hence the influence, of subtrees of X particles upon subtrees of Y particles. We begin by constructing space-partitioning trees for the X particles and Y points (see Section 2.3). The leaf nodes of these trees can contain multiple points. We also cache at each node in the X tree the maximum particle weight in the node ($w(X)$). At leaf nodes, we sort the particles in order of decreasing weight.

The algorithm proceeds by doing a depth-first recursion down the Y tree. For each node, we maintain a list of X nodes that could contain the best particle (candidates). We know the particle of maximum weight in a given node X. Thus, we can bound the influence of X by considering the cases when that particle is as close (or far) from Y as possible.

For each X node we compute the lower and upper bounds of the influence of the maximum particle in the node on all points in the Y node ($f^{\{l,u\}}$) by evaluating the kernel at the upper and lower bound on the distances to particles in the node ($d\{i, l\}$). The largest lower bound on influence is the *pruning threshold* (τ): any candidate node whose upper bound is less than this threshold cannot possibly contain the best particle, and hence need not be considered. See Figures 1 and 2 for an example.

In each recursive step, we choose one Y child on which to recurse. Initially, the set of X candidates is the set of candidates of the parent. We sort the candidates by lower bound, which allows us to explore the most promising nodes first. For each of the candidates' children, we compute the lower bound on distance and hence the upper bound on influence. Any candidates that have upper bound less than the pruning threshold are pruned. For those that are kept, the lower influence bound is computed; these nodes have the potential to become the new best candidate.

The influence bounds tighten as we descend the tree, allowing an increasingly number of nodes to be pruned. Once we reach the leaf nodes, we begin looking at individual particles. The candidate nodes are sorted by lower influence bound, and the particles are sorted by weight, so we examine the most promising particles first and minimize the number of individual particles examined. In many cases, we only have to examine the first few particles in the first node, since the pruning threshold often increases sufficiently to prune the remaining candidate nodes. Figures 3 and 4 contain pseudo-code for the algorithm.

For a given node in the Y tree, the list of candidate nodes in the X tree is valid for all the points within the Y node, which is the secret behind the efficiency of dual-tree recursion. In this way, pruning decisions are shared among Y points when possible.

2.3 Spatial indices

Spatial indices (sometimes called *spatial access methods*) intelligently subdivide a set into regions of high locality given some concept of distance. We briefly review two commonly-used spatial indices.

2.3.1 Kd-trees

A kd-tree operates on a vector field, and recursively chooses a dimension and split point to localize particles The dimension of largest spread is typically chosen as splitting dimension. Kd-trees are effective in low dimensional settings; a 2-D example is given in figure 1 (not all levels are shown).

2.3.2 Anchors hierarchy and metric trees

Metric trees are more relaxed in their requirements than kd-trees; they need only a defined distance metric. Nodes in a metric tree consist of a *pivot* (a point lying at the centre of the node), and *radius*. All points belonging to the node must have a distance to the pivot smaller than the radius of the node.[5] The *Anchors hierarchy* was introduced by Moore in [11] and is an efficient means of constructing a metric tree. Unlike kd-trees, metric tree construction and access costs do not have factors that explicitly depend on dimension.

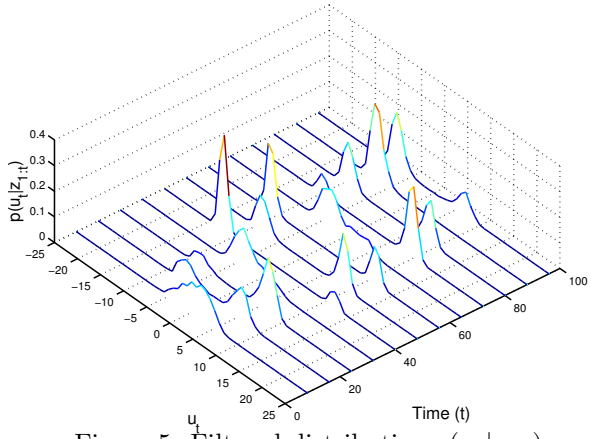


Figure 5: Filtered distribution $p(\mathbf{u}_t|\mathbf{z}_{1:t})$

3 Performance in N

We turn to empirical evaluation of the dual-tree algorithm. In this section, we focus on performance in synthetic and real-world settings as N grows; comparisons are made both in settings where the distance transform is applicable and where it is not. We present results in terms of both CPU time and number of distance computations (kernel evaluations) performed. This is important as in some applications the kernel evaluation is extremely expensive and thus dominates the runtime of the algorithm.

[5]Note: it is not the case that all points within the radius of the pivot belong to the node.

3.1 Multi-modal non-linear time series

Consider the following standard reference model [3, 6]:

$$u_{t+1} = \frac{1}{2}u_t + 25\frac{u_t}{1+u_t^2} + 8\cos 1.2t + v_{t+1}$$

$$z_{t+1} = \frac{u_{t+1}^2}{20} + w_{t+1}$$

where $v_t \sim \mathcal{N}(0, \sigma_v)$ and $w_t \sim \mathcal{N}(0, \sigma_w)$. The filtered distribution is bimodal and highly non-linear (figure 5), meaning the standard particle filter produces significant error even with a high particle count. After running a standard SIR particle filter, we implemented the MAP sequence estimation described in Section 1.2. Figure 6 demonstrates the accuracy gained by calculating the MAP solution. We chose a one-dimensional setting so that the dual-tree algorithm could be directly compared against the distance transform (using the modified algorithm from Section 2.1.1). Figures 7 and 8 summarize the results. It is clear that the distance transform is superior in this setting, although the dual-tree algorithm is still quite usable, being several orders of magnitude faster than the naïve method.

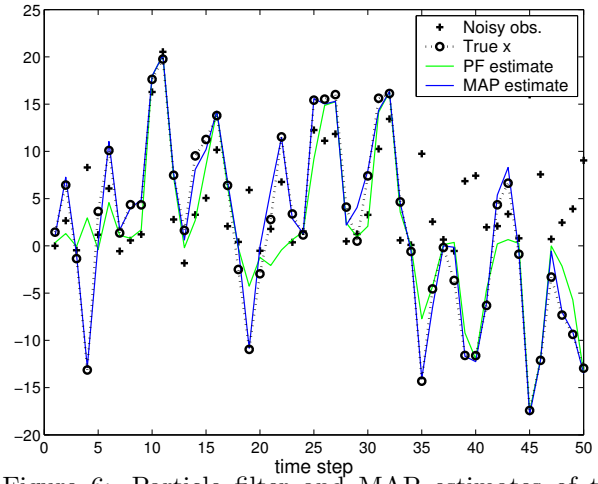


Figure 6: Particle filter and MAP estimates of the latent state in the 1-D time series experiment. Mean error for the particle filter was 4.05, while the MAP solution achieved a mean error of 1.74.

3.2 Beat-tracking

Beat-tracking is the process of determining the time slices in a raw song file that correspond to musical beats. This is a challenging problem: both the tempo and phase of the beats must be estimated throughout

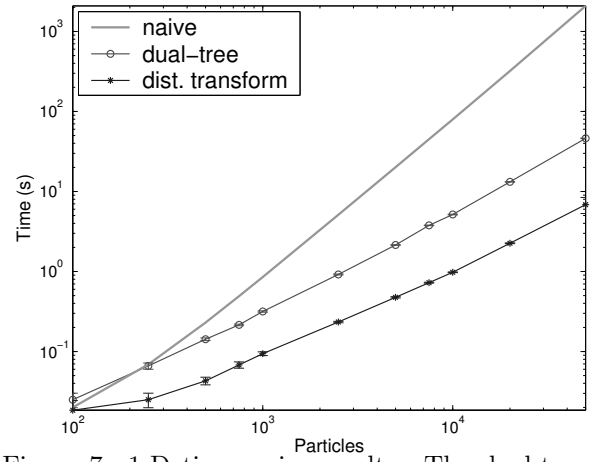


Figure 7: 1-D time series results. The dual-tree algorithm became more efficient than naïve computation after approximately 70ms of compute time. Both dual-tree and distance transform methods show similar asymptotic growth, although the constants in the distance transform are approximately three times smaller.

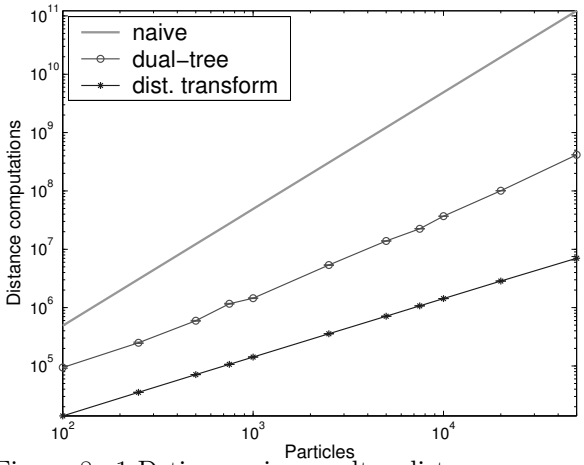


Figure 8: 1-D time series results: distance computations v. particle count.

the song. MAP sequence estimation after particle filtering has achieved impressive results in the literature. We omit the details of the probability model for the sake of brevity, but a full explanation is found in [10]. The algorithm used was the forward pass particle filter, backward pass Viterbi algorithm described in Section 1.2. Since the state space of this model is a three-dimensional Monte Carlo grid, the distance transform cannot be used. Figures 9 and 10 summarize the results: songs can be processed in seconds rather than hours with this method. Using the fast method also enables more particles to be used, which results in a better solution: the probability of the MAP sequence with 50000 particles was $p = 0.87$, while using 1000 particles resulted in a MAP sequence of probability $p = 0.53$.

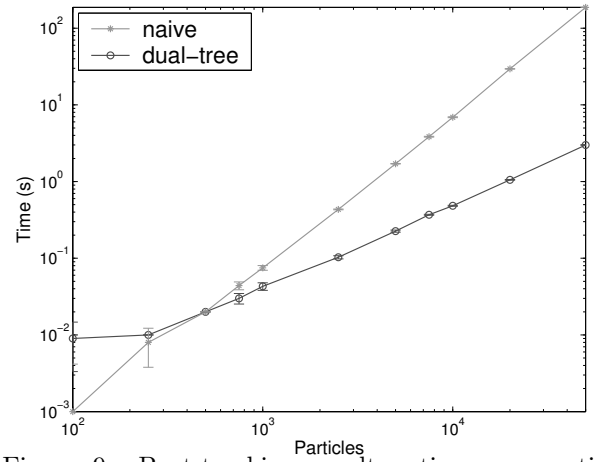


Figure 9: Beat-tracking results: time v. particle count. The dual-tree method becomes more efficient at $t = 10$ms, and thereafter dominates the naïve method.

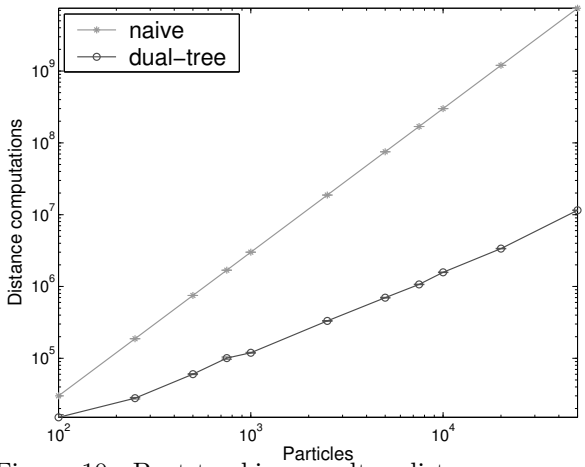


Figure 10: Beat-tracking results: distance computations v. particle count.

4 The effect of other parameters

4.1 Distribution and dimensionality

To examine the effects of other parameters on the behaviour of the dual-tree algorithm, we ran several experiments varying dimensionality, distribution, and spatial index while keeping N constant. We used two spatial indices: kd-trees and metric trees (built using the Anchors hierarchy) as described in Section 2.3. We generated synthetic data by drawing points from a mixture of Gaussians distributed evenly in the space. Figure 11 shows a typical clustered data distribution. In all runs the number of particles was held constant at $N = 20,000$, and the dimension was varied to a maximum of $d = 40$. Figures 12 and 13 show the results for CPU time and distance computations, respectively. In these figures, the solid line represents a uniform distribution of particles, the dashed line represents a 4-cluster distribution, the dash-dot line has

20 clusters, and the dotted line has 100. We expect methods based on spatial indexing to fare poorly given uniform data since the average distance between points quickly becomes a very peaked distribution in high dimension, reducing the value of distance as a measure of contrast. The results are consistent with this expectation: for uniform data, both the kd-tree and anchors methods exceeded $O(N^2)$ distance computations when $d \geq 12$. More surprising is that the kd-tree method consistently outperformed the anchors method on uniform data even up to $d = 40$. The depth of a balanced binary kd-tree of 20000 particles and leaf size 25 is ten, so for $d > 10$ there are many dimensions that are not split even a single time!

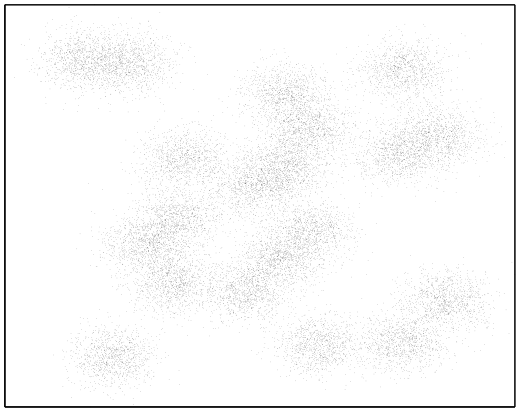


Figure 11: Synthetic data set with $c = 20$ clusters.

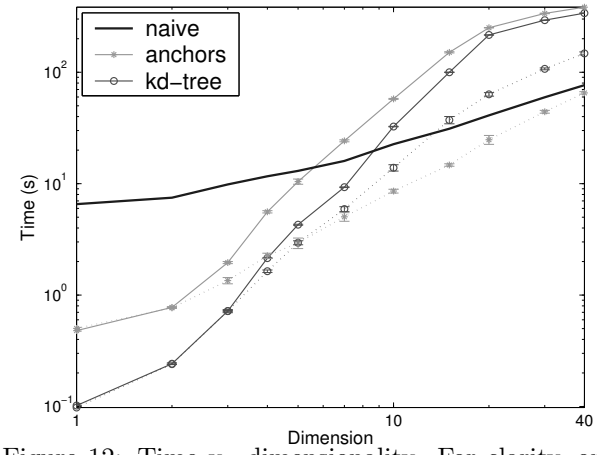


Figure 12: Time v. dimensionality. For clarity, only the uniform distribution and one level of clustered data are shown. This experiment demonstrates that some structure is required to accelerate max-kernel in high dimensions.

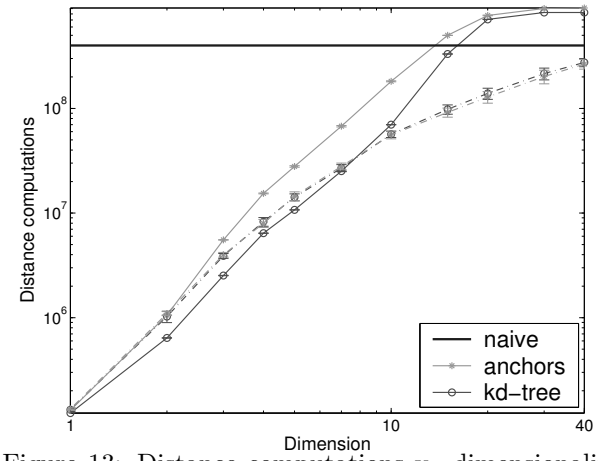


Figure 13: Distance computations v. dimensionality. The level of clustering shown is less than in figure 12. For kd-trees, clustering hurts performance when $d \leq 8$.

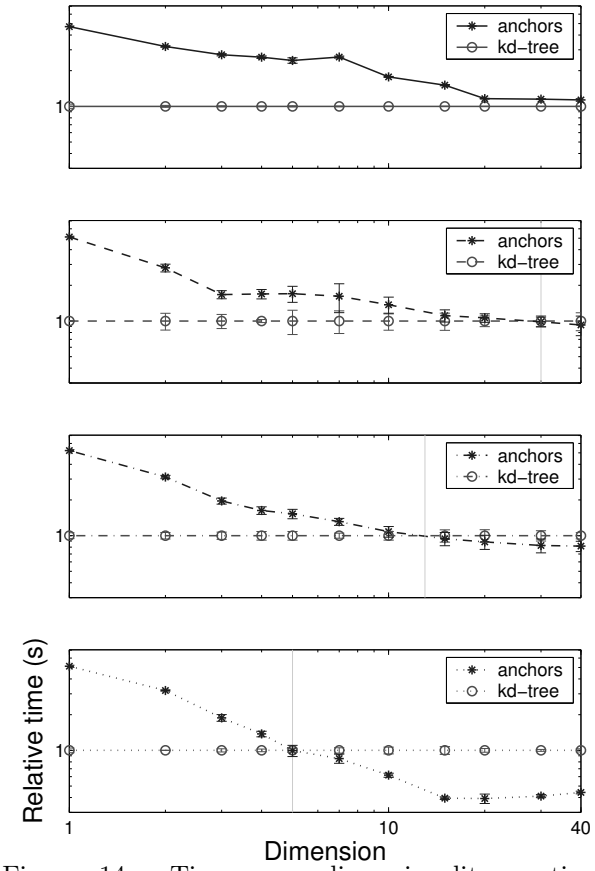


Figure 14: Time v. dimensionality; ratio to kd-tree = 1. Metric trees are better able to properly index clusters: the more clustered the data, the smaller dimensionality required for the anchors method to outperform kd-trees ($d = 30$ for somewhat-clustered data, $d = 15$ for moderately-clustered data, and $d = 6$ for significantly-clustered data).

Of more practical interest are the results for clustered data. It is clear that the distribution vastly affects the runtime of dual-tree algorithms; at $d = 20$, performing max-kernel with the anchors method was six times faster on clustered data compared to uniform. We expect this effect to be even greater on real data sets, as the clustering should exist on many scales rather than

simply on the top level as is the case with our synthetic data. It is also interesting to note the different effect that clustering had on kd-trees compared to metric trees. For the anchors method, clustering always improved the runtime, albeit by a marginal amount in low dimensions. For kd-trees, clustering *hurt* performance in low dimensions, only providing gains after about $d = 8$. The difference in the two methods is shown in figure 14.

4.2 Effect of kernel width

To measure the effect of different kernels, we test both methods on a 1-D uniform distribution of $200,000$ points, and use a Gaussian kernel with bandwidth (σ) varying over several orders of magnitude. The number of distance computations required was reduced by an order of magnitude over this range (figure 15). Wider kernels allow the weights to have more contrast, hence affording more opportunities for pruning.

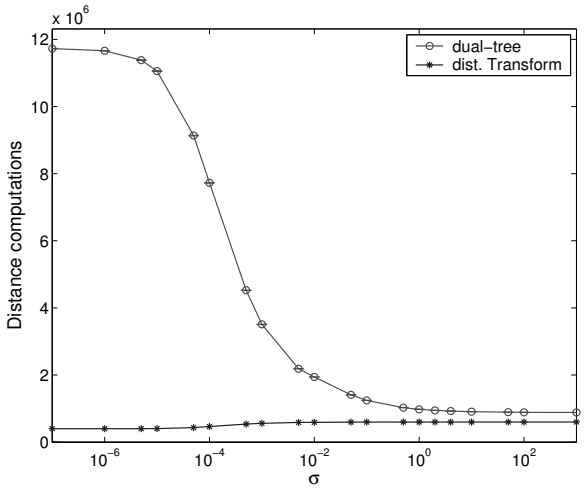


Figure 15: Effect of kernel choice: distance computations v. bandwidth of a Gaussian kernel.

5 Conclusion

Weighted maximum-kernel problems are common in statistical inference, being used, for instance, in belief propagation and MAP particle filter sequence estimation. We develop an exact algorithm based on dual-tree recursion that substantially reduces the computational burden of this procedure for a wide variety of kernels. It is particularly important when the state space lies on a multi-dimensional Monte Carlo grid, where, to our knowledge, no existing acceleration methods can be applied.

The method we present speeds up the inner loop of belief propagation, which means that it can be combined with other acceleration methods such as node pruning and dynamic quantization [2] to achieve even faster results, albeit at the expense of the loss of accuracy that those methods entail. The techniques we present could also be integrated seamlessly into hierarchical BP [4].

We also look at the other variables that affect the performance of dual-tree recursion, such as dimensionality, data distribution, spatial index, and kernel. These parameters have dramatic effects on the runtime of the algorithm, and our results suggest that more exploration is warranted into these effects—behaviour as N varies is only a small part of the story.

References

[1] N Bergman, *Recursive Bayesian estimation: Navigation and tracking applications*, Ph.D. thesis, Department of Electrical Engineering, Linköping University, Sweeden, 1999.

[2] J M Coughlan and H Shen, *Shape matching with belief propagation: Using dynamic quantization to accomodate occlusion and clutter*, GMBV, 2004.

[3] A Doucet, N de Freitas, and N J Gordon (eds.), *Sequential Monte Carlo methods in practice*, Springer-Verlag, 2001.

[4] P Felzenszwalb and D Huttenlocher, *Efficient belief propagation for early vision*, CVPR, 2004.

[5] P Felzenszwalb, D Huttenlocher, and J Kleinberg, *Fast algorithms for large-state-space HMMs with applications to web usage analysis*, NIPS (2003).

[6] S J Godsill, A Doucet, and M West, *Maximum a posteriori sequence estimation using Monte Carlo particle filters*, Ann. Inst. Stat. Math. **53** (2001), no. 1, 82–96.

[7] A Gray and A Moore, *'N-Body' problems in statistical learning*, NIPS, 2000, pp. 521–527.

[8] A Gray and A Moore, *Rapid evaluation of multiple density models*, AISTATS, 2003.

[9] A T Ihler, E B Sudderth, W T Freeman, and A S Willsky, *Efficient multiscale sampling from products of Gaussian mixtures*, NIPS 16, 2003.

[10] D Lang and N de Freitas, *Beat tracking the graphical model way*, NIPS 17, 2004.

[11] A Moore, *The Anchors Hierarchy: Using the triangle inequality to survive high dimensional data*, UAI 12, 2000, pp. 397–405.

[12] J Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan-Kaufmann, 1988.

[13] C P Robert and G Casella, *Monte Carlo statistical methods*, Springer-Verlag, New York, 1999.

Generative Model for Layers of Appearance and Deformation

Anitha Kannan[1], **Nebojsa Jojic**[2], **Brendan J. Frey**[1]
[1] Probabilistic and Statistical Inference Group, University of Toronto
[2] Microsoft Research, Redmond, WA, USA

Abstract

We are interested in learning generative models of objects that can be used in wide range of tasks such as video summarization, image segmentation and frame interpolation. Learning object-based appearance/shape models and estimating motion fields (deformation field) are highly interdependent problems. At the extreme, all motions can be represented as an excessively large set of appearance exemplars. However, a more efficient representation of a video sequence would save on frame description if it described the motion from the previous frame instead. The extreme in this direction is also problematic as there are usually causes of appearance variability other than motion. The flexible sprite model (Jojic and Frey 2001) illustrates the benefits of joint modelling of motion, shape and appearance using very simple models. The advantage of such a model is that each part of the model tries to capture some of the variability in the data until *all* the variability is decomposed and explained through either appearance, shape or transformation changes. Yet, the set of motions modelled is very limited, and the residual motion is simply captured in the variance maps of the sprites. In this paper, we develop a better balance between the transformation and appearance model by explicitly modelling arbitrary large, non-uniform motion.

1 Introduction

Our objective is to learn generative models of objects in a visual scene so that scene analysis (such as video summarization) can be efficiently performed. An important component of scene analysis involves learning object based appearance/shape models and estimate motion reliably. These two interesting problems of learning object based appearances and estimating motion are extensively studied separately even though both appearance and motion provide independent cues for estimating each other. In this work, we introduce a probabilistic generative model that unifies appearance modelling and motion estimation.

A step in this direction is reported in (Jojic and Frey 2001), as a layered extension of (Frey and Jojic 1999) for multiple objects. Here, the goal is to learn a layered density model for image formation. Given an input video sequence, the approach iteratively updates the appearances and masks of objects associated with each layer and the estimates of *global transformation(motion)* of the objects while capturing the residual motion in the variance maps of the sprites. Despite this appearance flexibility, the model requires an excessive number of appearance classes to capture many types of nonuniform large motions for which the translational variable is not a sufficient descriptor. We describe a new generative model for layered image formation that simultaneously learns deformation-invariant appearances and infer complex deformation fields. We use variational inference and generalized EM for learning and present results on flow computation, image segmentation and frame interpolation.

2 Related work

In a scene with multiple objects, approaches to motion estimation that operate on *matching* patches from one image to another (Lucas and Kanade 1981) under perform at the boundary regions due to occlusions and disocclusions. A good appearance model enables effective handling of boundary regions. On the other hand, objects can undergo complex deformations, and motion provides useful cues to learn appearances (Black and Jepson 1996). Thus, the estimation of appearances and motion should be done in tandem. A popular approach to this is to use a layered representation

in which we decompose a 3 dimensional scene into a set of 2 dimensional layers.

One such layered formalism is based on mixture models (Ayer and Sawhney 1995), (Jepson and Black 1993), (Weiss and Adelson 1996) in which each pixel is assigned probabilistically to one of several layers. When multiple objects are moving in a scene, there is a fair amount of occlusions and disocclusions, and without proper appearance models for the objects in the scene, it is extremely difficult to find the boundaries of the object.

In (Black and Fleet 2000), a framework for modelling motion discontinuities is presented. In this work, the foreground and background are separated by a straight edge within a single, fixed window in the image sequences. The image sequence within the window is modelled by a generative model that predicts the image at time t from the image at time $t-1$ using unknown state variables that describe the location of the edge and the motions of the foreground and background. An algorithm based on particle filtering is used to infer the location of the edge, motion vectors for the foreground and the background at each time step. Again, this approach does not have explicit model for appearances of the objects, but instead relies on straight edge to differentiate foreground and background pixels within a small window. Moreover, for complex object shapes, a single edge may not be sufficient to differentiate the two layers.

In (Jojic and Frey 2001), a generative model framework is used to automatically learn layers of "flexible sprites", which are probabilistic 2-dimensional appearance maps and masks of moving, occluding objects. An important assumption of this model is that pixels belonging to a sprite move with the same velocity (for instance, uniform translation). For many interesting video sequences, this assumption is too rigid.

In (Frey, Jojic and Kannan 2003), we suggest linearizing the transformation manifold locally. This approach has two drawbacks - it requires an additional global transformation for finding the position and often a linear manifold is not sufficient to capture *large* complex deformation. The use of low-frequency wavelets for smooth deformation fields (Jojic et al. 2001) suffers from the same problem.

3 Flexible sprites with deformation fields

Fig. 1 shows the hierarchical generative model that describes the process involved in two-layer image formation. The statistical generative process is as follows: For each layer, an appearance and a mask are generated from appropriate prior distributions associated with object classes. We sample deformation vectors for each pixel. The deformation field is then applied to both the appearance and the mask. The position variables are randomly selected and the appropriate latent images shifted in accordance. The final image is composed from the layers according to the masks, which can be either continuous or discrete.

At this juncture, we would like to point out that the deformation field is fully expressive and nonlinear, and the model without the position variable can still capture well the correct global motion. We have added the shift variable only to serve the purposes of regularization and speedup of computation. There are too many relatively good solutions to arbitrary matching patches in the mean image and the observation. The existence of the shift variable limits the search space for the deformation field estimation, *and* regularizes the search.

This is also the main difference from our earlier work (Frey, Jojic and Kannan 2003), where we used a linearizing manifold locally. To make this linearization work, an added nonlinearity is needed, and for that purpose, we used discrete shifts.

4 A generative model for occluded patches in motion

Although the following discussion applies to an arbitrary number of layers, we consider for simplicity a two-layer model, consisting of a foreground and a background. We treat foreground appearance (denoted by $\mathbf{f}$) and background appearance ($\mathbf{b}$) as parameters that apply to an entire sequence. (In the full model, there are several layers and several appearance classes that can occupy them).

We associate with the foreground layer a binary mask $\mathbf{m}$ of the same size as $\mathbf{f}$ such that $\mathbf{m}_i$=1 indicates that the corresponding pixel is a foreground pixel. Let the probability that $\mathbf{m}_i = 1$ be α_i so that

$$P(\mathbf{m}) = \prod_i \alpha_i^{m_i}(1-\alpha_i)^{(1-m_i)}.$$

Although the multiplicative alpha map we used in some of our previous papers is attractive for modeling mixed pixels, the binary mask tends to allow for a more robust inference (Frey and Jojic 2004)(Williams and Titsias 2003).

Each pixel in the latent images undergoes a deformation. In this paper, we use a discrete coordinate system for clarity, although techniques for sub-pixel inference and multi-scale search can be used. A priori the foreground and background motion vectors, $\mathbf{U}$ and $\mathbf{V}$ are independent and follow uniform distribution denoted by $P(\mathbf{U})$ and $P(\mathbf{V})$. We can favor smaller motions by using, for instance, a Gaussian prior on displacement.

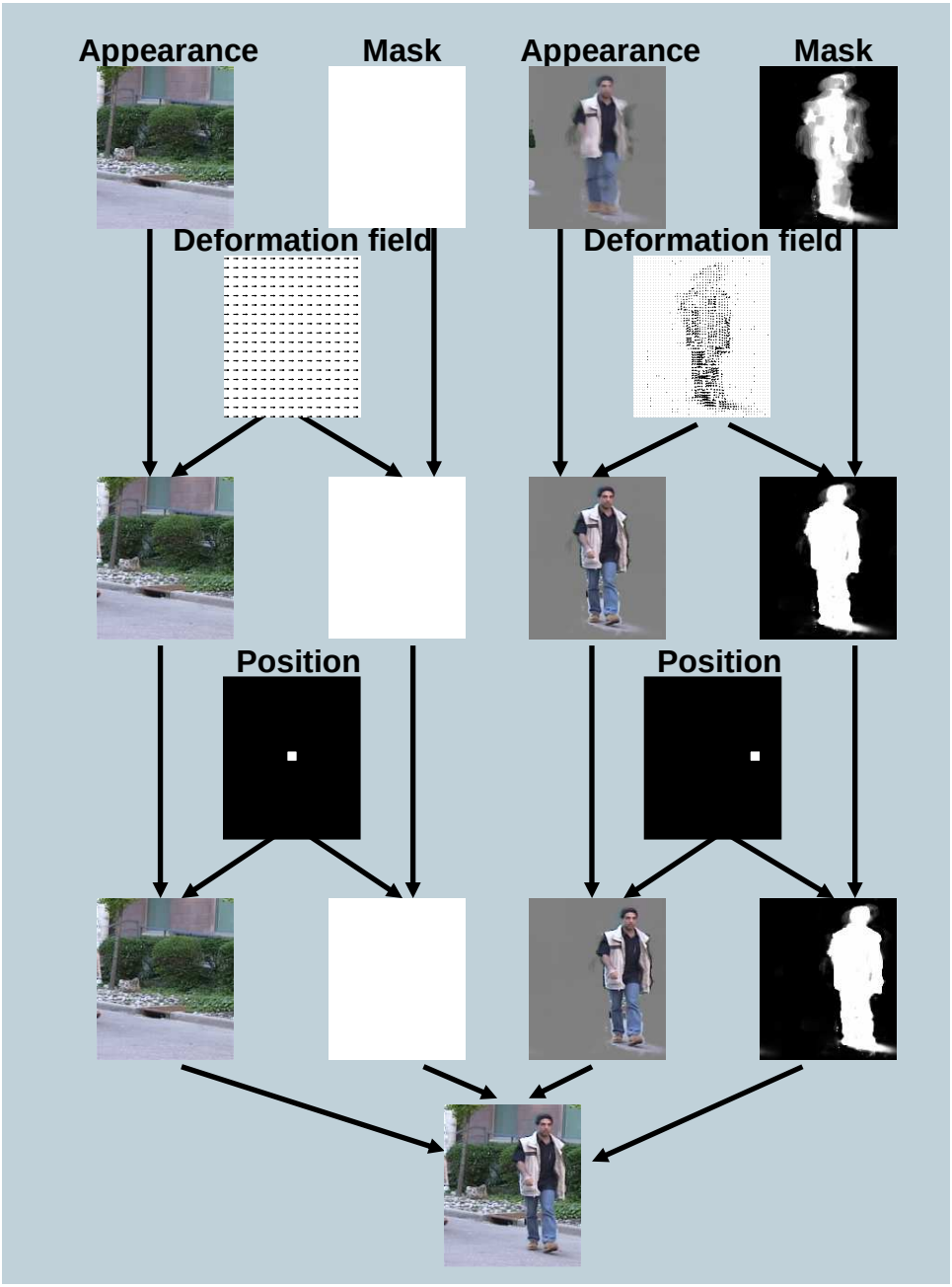


Figure 1: Illustration of the generative model using two layers. The images used in this figure are obtained by learning the model using a video sequence of a person walking towards the camera diagonally. The deformation model is nonlinear and is fully expressive; the additional level of global transformation provides regularization and offers significant computational advantage as discussed in sec. 3, but could be instead absorbed into the deformation field.

Every $M \times N$ observed frame is decomposed into a $(M-K+1) \times (N-K+1)$ grid of $K \times K$ *overlapping* patches in the spirit similar to our epitomic representations (Jojic, Frey and Kannan 2003). We let $\mathcal{P}(\mathbf{z})$ denote the set of coordinates that are in the patch centered at $\mathbf{z}$ so that $\mathcal{P}(\mathbf{z}) = \{\mathbf{w} : |\mathbf{w} - \mathbf{z}| \leq K\}$ and the corresponding pixel intensities to be $I(\mathcal{P}(\mathbf{z}))$

For pixel $I^t(\mathbf{z})$ in the observed image at time t, the foreground motion vector is represented by the random variable $\mathbf{U}^t(\mathbf{z})$, and the background motion vector by the random variable $\mathbf{V}^t(\mathbf{z})$.

To generate a pixel $I^t(\mathbf{z})$ in the observed image at time t, a foreground motion vector , $\mathbf{U}^t(\mathbf{z}) = \mathbf{u}$ from $P(\mathbf{U})$ and a background motion vector $\mathbf{V}^t(\mathbf{z}) = \mathbf{v}$ from $P(\mathbf{V})$ are drawn. The intensity of the pixel in the patch $\mathcal{P}(\mathbf{z})$ at time t is generated using :

$$I^t(\mathbf{w} \in \mathcal{P}(\mathbf{z})) = \mathbf{f}(\mathbf{w}+\mathbf{u})^{\mathbf{m}(\mathbf{w}+\mathbf{u})} * \mathbf{b}(\mathbf{w}+\mathbf{v})^{(1-\mathbf{m}(\mathbf{w}+\mathbf{u}))} + noise$$

Thus, when $\mathbf{m}(\mathbf{w}+\mathbf{u}) = 1$, foreground pixel intensity is observed, and when $\mathbf{m}(\mathbf{w}+\mathbf{u}) = 0$, background pixel intensity is observed at pixel location $\mathbf{w} \in \mathcal{P}(\mathbf{z})$. We assume that the (sensor) noise is Gaussian with variance σ^2 so that the observation likelihood of the patch is a Gaussian given by,

$$P(I^t(\mathcal{P}(\mathbf{z}))|\mathbf{U}^t(\mathbf{z})=\mathbf{u}, \mathbf{V}^t(\mathbf{z})=\mathbf{v}) \propto exp\Big[- \sum_{\mathbf{w} \in \mathcal{P}(\mathbf{z})} \frac{(\mathbf{f}(\mathbf{w}+\mathbf{u})^{\mathbf{m}(\mathbf{w}+\mathbf{u})}\mathbf{b}(\mathbf{w}+\mathbf{v})^{1-\mathbf{m}(\mathbf{w}+\mathbf{u})} - I^t(\mathbf{w}))^2}{2\sigma^2} \Big]$$

As in the epitome representation, we assume that the patch appearances are independent.

Let the motion fields in all nearby frames be $\mathcal{U}$ and $\mathcal{V}$ and the observed patches in all nearby frames be $\mathcal{I}$ so that joint distribution is proportional to

$$P(\mathcal{U}, \mathcal{V}, \mathcal{I}, \mathbf{m}) \propto \prod_t \prod_{\mathbf{z}} P(\mathbf{m}) P(I^t(\mathcal{P}(\mathbf{z}))|\mathbf{U}^t(\mathbf{z})=\mathbf{u}, \mathbf{V}^t(\mathbf{z})=\mathbf{v}, \mathbf{m}) \quad (1)$$

5 Inference & Learning

For learning, the natural choice is the Expectation Maximization algorithm (Dempster, Laird and Rubin 1977) that maximizes the likelihood of observation. However, as exact inference is intractable, we resort to variational approximation (Jordan et al. 1999) for the posterior and use generalized EM (Neal and Hinton 1998) for learning.

For each observed image, we approximate posterior as:

$$P(\mathbf{U}, \mathbf{V}, \mathbf{m}|I^t) = \prod_{\mathbf{z}} q^t(\mathbf{U}(\mathbf{z}), \mathbf{V}(\mathbf{z})) q^t(\mathbf{m})$$

Letting β_i^t be the probability that $\mathbf{m}_i$=1 given the i^{th} pixel in the t^{th} frame, and $\overline{\beta_i^t}$ = 1-β_i^t,

$$q^t(\mathbf{m}) = \prod_i (\beta_i^t)^{\mathbf{m}_i} \overline{\beta_i^t}^{(1-\mathbf{m}_i)}$$

Generalized EM maximizes the bound on the (log) probability of the data:

$$logP(\mathcal{I}) \geq \sum_t \sum_{\mathbf{z}} \sum_m \sum_{u,v} q^t(\mathbf{U}(\mathbf{z}), \mathbf{V}(\mathbf{z})) q^t(\mathbf{m}) log \frac{P(U(\mathbf{z}), V(\mathbf{z}), I^t(\mathcal{P}(\mathbf{z})), \mathbf{m})}{q^t(\mathbf{U}(\mathbf{z}), \mathbf{V}(\mathbf{z})) q^t(\mathbf{m})}$$

Before, we derive the update equations, we define two quantities: We allow every patch to shift by at most D pixels. This reduces the search space and therefore the computational cost. When there is large motion, we can further reduce this search space D by incorporating global transformation, as described in sec. 3.

The set of all coordinates in the observed image whose $K \times K$ patches can "reach" coordinate $\mathbf{x}$ in $\mathbf{f}$ or $\mathbf{b}$ when moved by at most D is $\mathcal{R}(\mathbf{x}) = \{\mathbf{z} : |\mathbf{x} - \mathbf{z}| \leq (K-1)/2 + D\}$ The set of all motion vectors for the patch at $\mathbf{z}$ in observed image that cause a pixel in the patch to be mapped to $\mathbf{x}$ in $\mathbf{f}$ or $\mathbf{b}$ is

$$\mathcal{M}(\mathbf{x}, \mathbf{z}) = \{\mathbf{u} : |(\mathbf{x} - \mathbf{z}) - \mathbf{u}| \leq (K-1)/2; |\mathbf{u}| \leq D\}.$$

The posterior distribution over the motion vectors is

$$q^t(\mathbf{U}(\mathbf{z})=\mathbf{u}, \mathbf{V}(\mathbf{z})=\mathbf{v}) = \rho \exp\Big[-\frac{1}{2\sigma^2} \sum_{\mathbf{w} \in \mathcal{P}(\mathbf{z})} \Big\{ \beta_{\mathbf{w}}^t (I^t(\mathbf{w}) - \mathbf{f}(\mathbf{w}+\mathbf{u}))^2 + \overline{\beta_{\mathbf{w}}^t} (I^t(\mathbf{w}) - \mathbf{b}(\mathbf{w}+\mathbf{v}))^2 \Big\} \Big] \quad (2)$$

where ρ ensures that $\sum_{\mathbf{u}} \sum_{\mathbf{v}} P(\mathbf{U}^t(\mathbf{z}) = \mathbf{u}, \mathbf{V}^t(\mathbf{z}) = \mathbf{v}|I^t) = 1$. Due to the use of binary mask, the computation inside the exponential splits into sum of two distance measures. When the posterior over the mask is peaked, pixels in the observed patch that are attributed to foreground are compared with patch from the shifted foreground, and observed pixels that belong to background are matched with the corresponding shifted patch in the background. This distance computation need not be done on a patch by patch basis, but instead by observing that each pixel participates in a large number of patches, we can employ simple trick using cumulative sums and calculate the distances for all patches in tandem.

The posterior distribution over the mask is

$$\beta_{\mathbf{w}}^t = 1/\Big[1 + exp \sum_{\mathbf{u},\mathbf{v}} q(\mathbf{U}^t(\mathbf{z})=\mathbf{u}, \mathbf{V}^t(\mathbf{z})=\mathbf{v}) ((I^t(\mathbf{w}) - \mathbf{b}(\mathbf{w}+\mathbf{v}))^2 - (I^t(\mathbf{w}) - \mathbf{f}(\mathbf{w}+\mathbf{u}))^2) \Big] \quad (3)$$

Figure 2: Top row: entire sequence of 6 frames used to train the model. Notice that one person is moving towards the camera, inducing a zooming in effect, and another person moving in the background. Bottom row: interpolated frame between adjacent frames in the top row. Interpolation is performed using the learned parameters and the inferred deformation fields.

We use $< . >= \sum_t \sum_{\mathbf{z}\in\mathcal{R}(\mathbf{y})} \sum_{\mathbf{u}\in\mathcal{M}(\mathbf{y},\mathbf{z})} \sum_{\mathbf{v}\in\mathcal{M}(\mathbf{y},\mathbf{z})}$ to represent the sufficient statistic collected from all pixels, $\mathbf{z}$ in all frames, t, that map pixel $\mathbf{y}$ and the motion vectors for $\mathbf{z}$ that cause the pixel to be mapped.

The update for background appearance is:

$$\mathbf{b}(\mathbf{y}) \quad \frac{\overline{\beta^t_{\mathbf{y}-\mathbf{v}}} q(\mathbf{U}^t(\mathbf{z})=\mathbf{u}, \mathbf{V}^t(\mathbf{z})=\mathbf{v}) I^t(\mathbf{y}-\mathbf{v})}{\overline{\beta^t_{\mathbf{y}-\mathbf{v}}} q(\mathbf{U}^t(\mathbf{z})=\mathbf{u}, \mathbf{V}^t(\mathbf{z})=\mathbf{v})} \tag{4}$$

The above update involves aligning the observed pixel with respect to the background using the posterior distribution over the motion vectors and then multiplying this with the posterior probability of the pixel belonging to the background. Since multiple patches from all the frames contribute to updating the same pixel in the background, the denominator normalizes for multiple counts.

The foreground appearance is updated similarly:

$$\mathbf{f}(\mathbf{y}) \quad \frac{\beta^t_{\mathbf{y}-\mathbf{u}} q(\mathbf{U}^t(\mathbf{z})=\mathbf{u}, \mathbf{V}^t(\mathbf{z})=\mathbf{v}) I^t(\mathbf{y}-\mathbf{u})}{\beta^t_{\mathbf{y}-\mathbf{u}} q(\mathbf{U}^t(\mathbf{z})=\mathbf{u}, \mathbf{V}^t(\mathbf{z})=\mathbf{v})} \tag{5}$$

The prior probability of a pixel to be from the foreground is given by:

$$\alpha_{\mathbf{y}} \quad \frac{\beta^t_{\mathbf{y}-\mathbf{u}} q(\mathbf{U}^t(\mathbf{z})=\mathbf{u}, \mathbf{V}^t(\mathbf{z})=\mathbf{v})}{q(\mathbf{U}^t(\mathbf{z})=\mathbf{u}, \mathbf{V}^t(\mathbf{z})=\mathbf{v})} \tag{6}$$

We initialize the appearance variables to a reference frame (usually the middle frame in the sequence) and let the prior distribution over the mask to be uniform. We iterate between finding the posterior over motion vectors and the posterior over the mask in the Estep, and updating the appearances and the prior distribution for the mask in the Mstep. This procedure enables inferring the layered optical flow in images. However, in the full flexible sprites model, the chosen variational factorization, when combined with the variational factorization of the shifts in the original flexible sprites paper leads to efficient inference and learning whose results are shown in Fig. 1. We omit the mathematical details for brevity.

6 Experimental results

6.1 Modelling complex deformation and appearances in two layers

For this experiment, we used 6 frames of 88×133 RGB sequence shown in fig. 2. The sequence has a person moving towards the camera in front of a moving background inducing a complex deformation field. There is also another person walking behind in the opposite direction. The translational motion of the background is due to camera shake.

We trained our foreground-background model on this sequence using 5×5 overlapping patches. For computational reasons, we restricted the search space for the foreground motion to be 4 pixels in both directions (81 possible directions). The background motion was restricted to 2 pixels in the horizontal direction. The state space of the posterior distribution over the motion field has a cardinality of 405 ($9\times9\times5\times1$). Upon investigation, we found that the posterior distribution is peaked at a few values. This fact can be used to address the storage issue during inference. In fact, in our experiments we store the distribution of only the top 20 motion directions.

In Fig. 3b, we show the learned appearances of the foreground and background, and the probability distribution of the binary mask. It is interesting to notice that the learned mask distribution has captured the person walking behind as part of the foreground. Also, some of the occluded background pixels are filled in. In fig. 3a, results of learning flexible sprites model without deformation is shown. Since, the flexible sprites

Figure 3: Top row: *Flexible sprites model* Background, foreground (masked) and transparency mask learned using the two layer sprites model on the data in Fig. 2. The complex deformation of the foreground object can't be modelled using this model.
Bottom row:*Proposed model* Learned background, foreground and probability values of binary mask learned. The foreground appearance is invariant to deformation, and the background has lesser number of occluded pixels, and the mask captures the person moving away from the camera as part of the foreground.

model assumes that the pixels belonging to a layer move with the same velocity it can not handle non-uniform motion. In fact, some of the foreground pixels are misclassified to be background pixels as the corresponding background pixels are always occluded.

Inference in this model also gives us the distribution over the motion vectors for each pixel and for each layer. Using this we can compute the expected motion for each pixel by averaging the foreground and background motion weighted by their posterior probabilities. The middle frame is considered as the reference frame for which the motion is set to be 0. Fig. 4 shows the inferred deformation field for each frame with respect to the reference frame. Note, however, that our motion field is defined with respect to the *derived* foreground and background appearances in $\mathbf{b}$ and $\mathbf{f}$ which have more disoccluded pixels than any frame.

The learned appearances and the inferred flow vectors can be used to perform video interpolation. In Fig. 2 we present 1 frame interpolation between adjacent frames. See the accompanying website for more interpolation results.

6.2 Modelling mixtures of complex deformation and appearances in two layers

Our model can easily be extended to incorporate multiple layers of moving objects with appearance of each layer modelled as a mixture model.

In this experiment, we present results for learning a two layer model where the foreground appearance is modelled using a Gaussian mixture with 2 classes. We also allow the latent variables (appearances and probability masks) to be bigger than the observed image so as to learn a panoramic background.

We learn the model using 10 RGB frames ($138 \times 148 \times 3$) sampled from a longer video sequence (Fig. 6a). Each frame consists of one of the two persons (modelled using different classes) moving towards the camera in front of a non-stationary background. Notice that the images include scale changes in appearance due to zoom, complex motion of hands and legs, wrinkles in the clothing, and large shifts in the position.

We used larger appearances and masks (138×178) than the observed frames. Referring to our model in fig. 1, we first train the model without incorporation of the deformation to obtain the global position variables in each layer for each frame. Once the global shifts are inferred, we fix them to learn the deformation field and the parameters of the model in tandem as outlined in Sec. 5

In fig. 5, the parameters of the learned model are shown. Frames corresponding to the first appearance class have pixels belonging to the background that are always occluded. However, these pixels are visible in some frames where the other appearance class is present. By jointly modelling all the frames, we are able to fill in for almost all the occluded pixels belonging to the background for any given frame. This is further shown in fig. 5a. If we had chosen to learn two separate models for the two classes, the background will not have all its corresponding pixels observed.

For the pixels belonging to the texture less pathway, the prior probability distribution over the mask is close to uniform (fig. 5c & e). This suggests that for those regions that do not have enough textural variations to group them as belonging to one of the two layers, it is at best to assign equal probability for either layer to explain them.

In fig. 6, we present inference results for some representative frames, shown in fig. 6a. Fig. 6b is the corresponding inferred deformation field shifted according to inferred global transformation. The flow vectors

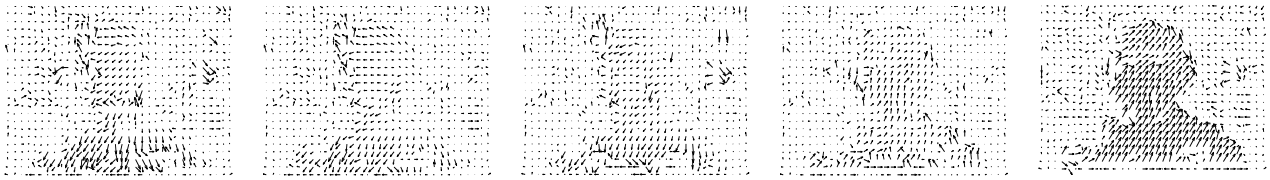


Figure 4: Inferred deformation field corresponding to the image sequence shown in Fig. 2(the flow field is drawn with reference to the fourth frame which is not shown here)

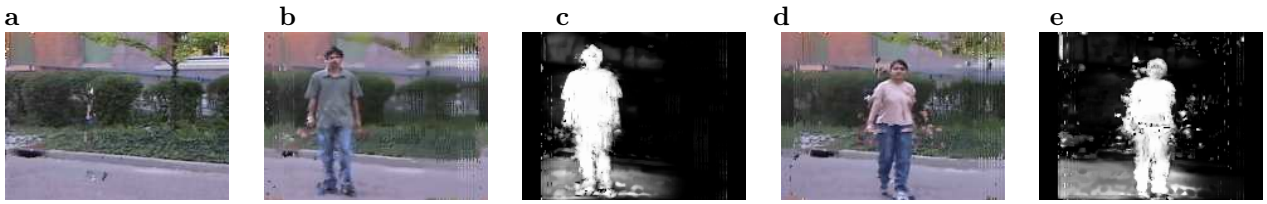


Figure 5: Parameters of two layered two class model learned using 10 frames from a video sequence (representative frames in fig. 6 a)Learned background is larger than the size of input image b) Appearance of foreground object of class 1 and c) the corresponding probability distribution of the binary mask (with white referring to probability of 1 for the pixel belonging to foreground) d) & e) appearance and probability mask of the second class of foreground layer.

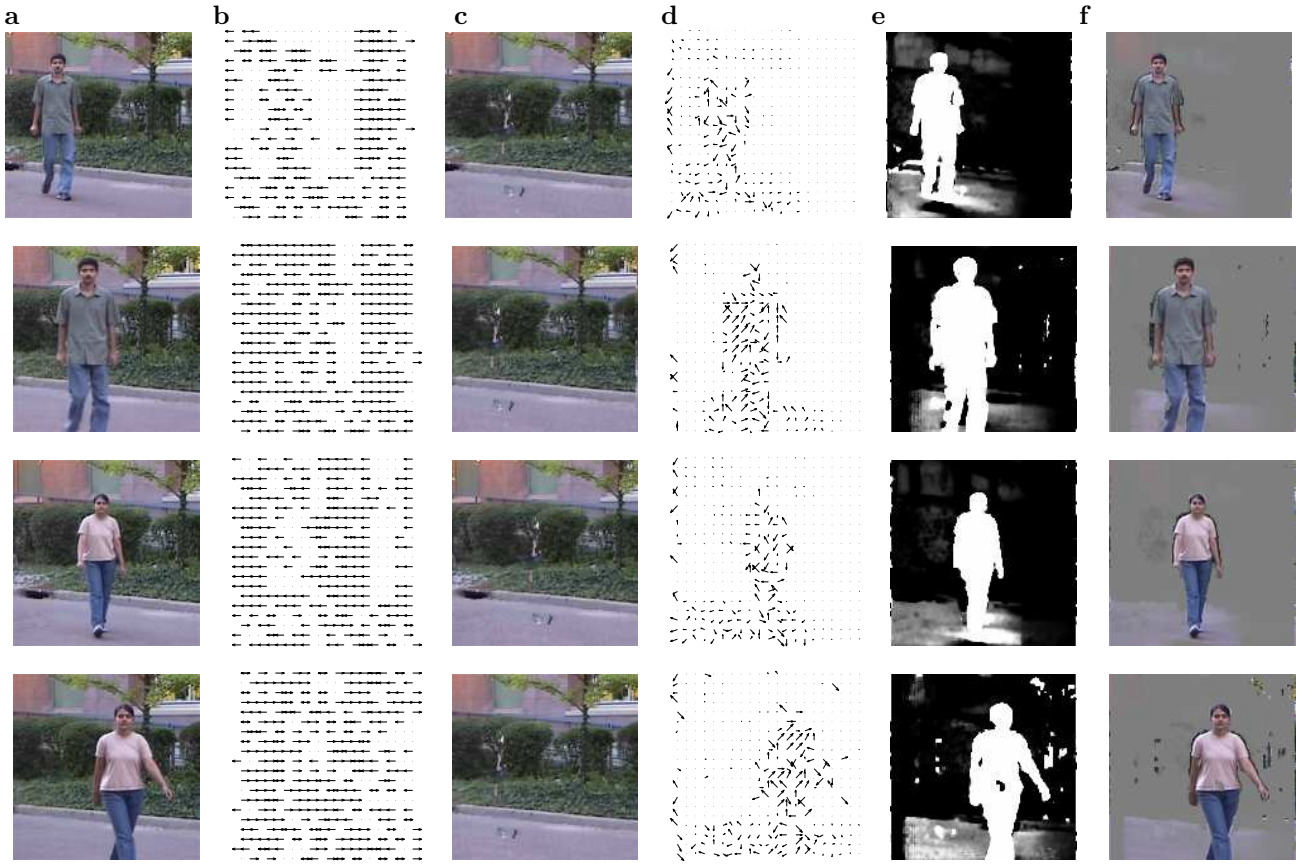


Figure 6: Illustration of inference for some frames of the sequence explained in sec. 6.2 a) frames from a sequence. b) Deformation field for the background. c) Deformed, globally transformed background. d) Deformation field for the foreground (masked). e) Distribution for the mask after global transformation is applied. f) Mask applied on the frame in a)

are drawn relative to the learned parameters. Each vector in this field represents the most probable (that vector that has the largest posterior probability) deformation vector for that pixel. The inferred deformation field for the foreground appearance is in fig. 6c. It is interesting to note that the deformation vectors for appearance are smoother and more consistent along the boundaries than on the central regions of the foreground object. As our approach learns a good appearance model, the boundaries of the objects are well defined but the motion within the object is not very coherent due to lack of enough texture variation between adjacent patches to reliably favor a particular motion direction. We contrast this with inferred flow vectors in the previous experiment (fig. 4) where we had enough textural variation in the central region of object of interest that we learned a much smoother flow field.

7 Conclusions

We have enriched the flexible sprites model with the deformable motion variables defined on overlapping patches. We assume that in each patch there exist two motion vectors and that some pixels are following one and others the other motion. The selection is defined by a patch of binary variables. These patches are also overlapping in the model of the mask, aligned with one of the layers. We were able to use this model of motion within the flexible sprites model and obtain better appearance, mask and motion estimates. See the web page http://www.psi.utoronto.ca/∼anitha/flex.html for additional results and videos.

Acknowledgments

The authors thank P.Anandan for discussion on the importance of combining top-down object appearance models with low-level visual cues, in particular, motion. We also thank Allan Jepson for his comments on an earlier version of the work.

References

Ayer, S. and Sawhney, H. S. 1995. Layered representation of motion video using robust maximum likelihood estimation of mixture models and mdl encoding. In *Proceedings of the International Conference on Computer Vision*, pages 777–784.

Black, M. J. and Fleet, D. J. 2000. Probabilistic detection and tracking of motion discontinuities. *International Journal on Computer Vision*.

Black, M. J. and Jepson, A. 1996. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Dempster, A. P., Laird, N. M., and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Proceedings of the Royal Statistical Society*, B-39:1–38.

Frey, B. and Jojic, N. 2004. Advances in algorithms for inference and learning in complex probability models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (*to appear*)*.

Frey, B. J. and Jojic, N. 1999. Estimating mixture models of images and inferring spatial transformations using the em algorithm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Frey, B. J., Jojic, N., and Kannan, A. 2003. Learning appearance and transparency manifolds of occluded objects in layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Jepson, A. and Black, M. J. 1993. Mixture models for optical flow computation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 760–761.

Jojic, N., Frey, B., and Kannan, A. 2003. Epitomic analysis of appearance and shape. In *Proceedings of International Conference in Computer Vision*. IEEE.

Jojic, N. and Frey, B. J. 2001. Learning flexible sprites in video layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Jojic, N., Simard, P., Frey, B. J., and Heckerman, D. 2001. Separating appearance from deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Jordan, M. I., Ghahramani, Z., Jaakkola, T., and Saul, L. 1999. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

Lucas, B. D. and Kanade, T. 1981. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Arti cial Intelligence*.

Neal, R. M. and Hinton, G. E. 1998. A view of the em algorithm that justi es incremental,sparse, and other variants. In Jordan, M. I., editor, *Learning in Graphical Models*. Kluwer Academic Publishers, Norwell MA.

Weiss, Y. and Adelson, E. 1996. A uni ed mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *Proceedings of IEEE Computer Vision and Pattern Recognition*.

Williams, C. W. and Titsias, M. K. 2003. Learning about multiple objects in images: Factorial learning without factorial search. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*. MIT Press, Cambridge MA.

Toward Question-Asking Machines: The Logic of Questions and the Inquiry Calculus

Kevin H. Knuth*
Computational Sciences Division
NASA Ames Research Center
Moffett Field, CA 94035-1000

Abstract

For over a century, the study of logic has focused on the algebra of logical statements. This work, first performed by George Boole, has led to the development of modern computers, and was shown by Richard T. Cox to be the foundation of Bayesian inference. Meanwhile the logic of questions has been much neglected. For our computing machines to be truly intelligent, they need to be able to ask *relevant questions*. In this paper I will show how the Boolean lattice of logical statements gives rise to the free distributive lattice of questions thus defining their algebra. Furthermore, there exists a quantity analogous to probability, called *relevance*, which quantifies the degree to which one question answers another. I will show that relevance is not only a natural generalization of information theory, but also forms its foundation.

1 INTRODUCTION

Intelligent machines need to actively acquire information, and the act of asking questions is central to this capability. Question-asking comes in many forms ranging from the simplest where an instrument continuously monitors data from a sensor, to the more complex where a rover must decide which instrument to deploy or measurement to take, and even the more human-like where a robot must verbally request information from an astronaut during in an in-orbit construction task.

Intelligence is not just about providing the correct solution to a problem. When vital information is lacking, intelligence is required to formulate relevant questions. For over 150 years mathematicians have studied the logic of statements (Boole, 1854); whereas the mathematics of questions has been almost entirely neglected. In this paper, I will describe my recent work performed in understanding the *algebra of questions*, and its associated calculus, the *inquiry calculus*.

Much of the material presented in this paper relies on the mathematics of partially-ordered sets and lattices. For this reason, I have included a *short appendix* to which the reader can refer for some of the mathematical background. Section §2 briefly discusses questions and the motivation for this work. Section §3 introduces the formal definition of a question. I develop the lattice of questions and its associated algebra in Section §4. I extend the question algebra to the inquiry calculus in Section §5 by introducing a bi-valuation called *relevance*, which quantifies the degree to which one question answers another. In section §6, I show that the inquiry calculus is not only a *natural* generalization of information theory, but also forms its foundation. Section §7 summarizes the results, discusses how information theory has been used for some time to address question-asking, and describes how this more general methodology and deeper understanding will facilitate this process.

2 QUESTIONS

Each and every one of us asks questions, and has done so since being able to construct simple sentences. Questions are an essential mechanism by which we obtain information, however as we all know, some questions are better than others. Questions are not always verbal requests, but are often asked in the form of physical manipulations or experiments: '*What happens when I let go of my cup of milk?*' or '*Will my mother make that face again if I drop it a second time?*' Questions may also be more fundamental, such as the saccade you make when you detect motion in your peripheral visual field. Or perhaps the issue is more effectively resolved by turning your head so as to deploy

*http://www.huginn.com/knuth/

both your visual and auditory sensory modalities. Regardless of their form, "questions are requests for information" (Caticha, 2004).

Many questions simply cannot be asked: there may be no one who will know the answer, no immediate way to ask it, you may not be allowed for a variety of reasons, or the question may be too expensive with respect to some cost criteria. In most situations, these questions cannot now be asked directly: '*Is there life in Europa's ocean?*', '*How fast does the SR71 Blackbird fly?*', '*How would radiation exposure on a Mars mission affect an astronaut's health?*', or '*What is the neutrino flux emitted from Alpha Centauri?*' In these cases, one must resort to asking other questions that do not directly request the information sought, yet still have relevance to the unresolved issue. This sets up the iterative process of inquiry and inference, which is essential to the process of learning—be it active learning by a machine, learning performed by a child, or the act of doing science by the scientific community.

Choosing relevant questions is a difficult task that requires intelligence. Anyone who has tried to perform a construction task with the assistance of a small child will appreciate this fact. Constantly being asked '*Do you need a hammer?*' by even the most enthusiastic helper can be a great annoyance when you are struggling to drill a hole. This is precisely the situation we will need to avoid when robots are used to assist us in difficult and dangerous construction tasks. Relevant questions asked by an intelligent assistant will be invaluable to minimizing risks and maximizing productivity in human-robot interactions. However, despite being an important activity on which we intelligent beings constantly rely, the mathematics of quantifying the relevance of a question to an outstanding issue has been surprisingly neglected.

3 DEFINING QUESTIONS

One of the most interesting facts about questions is that even though we don't know the answer, the question is essentially useless if we have absolutely no idea of what the answer could be. That is, when questions are asked intelligently, we already have a notion of the set of possible answers that the resolution may take. Richard T. Cox in his last paper captured this idea when he defined a question as the set of all logical statements that answer it (Cox, 1979).

The utility of such a definition becomes apparent when one considers the set of all possible answers to be a hypothesis space. The act of answering a question is equivalent to retrieving information, which will be used to further refine the probability density function over the hypothesis space, thereby reducing uncertainty. This can be formalized to a greater degree, and to our advantage, by realizing that a set of logical statements can be partially-ordered by the binary ordering relation '*implies*'. This set of logical statements along with its binary ordering relation $\rightarrow$, generically written in order-theoretic notation as $\leq$, forms a partially-ordered set, which can be shown to be a Boolean lattice (Birkhoff, 1967; Davey & Priestley, 2002). As a concrete example, consider a human-robotic cooperative construction task involving a robot named Bender and a human named Fry.[1] Bender has become aware that Fry will be in need of a tool, but must decide which tool Fry will prefer:

$$d = \textit{'Fry needs a drill!'}$$
$$w = \textit{'Fry needs a wrench!'}$$
$$h = \textit{'Fry needs a hammer!'}$$

These three atomic statements comprise the three mutually exclusive possibilities in Bender's hypothesis space. The Boolean lattice $\mathcal{A}$ (Figure 1), which I will interchangeably call the *statement lattice* or the *assertion lattice*, is the powerset of these three statements, formed by considering all possible logical disjunctions, ordered by the binary ordering relation '*implies*', $\rightarrow$. In an ideal situation, Bender's situational awareness would provide sufficient information to allow him to infer the tool Fry most probably needs. However, in reality, this will not always be the case, and Bender may need more information to adequately resolve the inference. The human way to accomplish this is to simply ask Fry for more information. Clearly, the most relevant question Bender can ask will depend both on the probabilities of the various hypotheses in this space, and on the specific issue Bender desires to resolve.

I now introduce a more formal definition of a question, which will allow us to generate a lattice of questions from a lattice of logical statements representing the hypothesis space. I first define a down-set (Davey & Priestley, 2002).

Definition 1 (Down-set) *A down-set is a subset J of an ordered set L where if $a \in J$, $x \in L$, $x \leq a$ then $x \in J$. Given an arbitrary subset K of L, we write the down-set formed from K as $J = \downarrow K = \{y \in L | \exists x \in K \text{ where } y \leq x\}$.*

Keep in mind that $\leq$ represents the ordering relation for the ordered set—in this case $\leq$ is equivalent to $\rightarrow$ for the lattice $\mathcal{A}$. A formalized version of Cox's definition of a question follows (Knuth, 2003a, 2004b, 2005).

[1] Bender and Fry are characters on the animated television series Futurama created by Matt Groening.

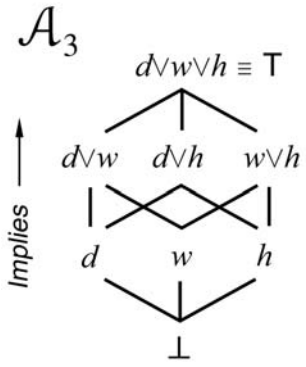


Figure 1: $\mathcal{A}_3$ is the Boolean lattice formed from three mutually exclusive assertions ordered by the relation '*implies*'. The bottom element $\bot$ is the absurdity, which is always false, and the top element $\top$ is the truism, which is always true.

Definition 2 (Question) *A question Q is defined as a down-set of logical statements $Q = \downarrow\{a_1, a_2, \ldots, a_n\}$. The question lattice $\mathcal{Q}$ is the set of down-sets of the assertion lattice $\mathcal{A}$ ordered by the usual set-inclusion $\subseteq$, so that $\mathcal{Q} = \mathcal{O}(\mathcal{A})$.*

This defines a question in terms of the set of statements that answer it, which includes all the statements that imply those statements. Note that I am using lowercase letters for assertions (logical statements), uppercase letters for questions (or sets), and script letters for ordered sets (lattices). The question lattice $\mathcal{Q}$ generated from the Bender's Boolean assertion lattice $\mathcal{A}$ is shown in Figure 2 with the following notation:

$$\begin{aligned} H &= \downarrow h = \{h, \bot\} \\ WH &= \downarrow w \vee h = \{w \vee h, w, h, \bot\} \\ DWH &= \downarrow d \vee w \vee h = \{d \vee w \vee h, \ldots\} \end{aligned}$$

This lattice shows all the possible questions that one can ask concerning the hypothesis space $\mathcal{A}$. For example, the question $H \cup DW$ is the set union of the questions H and DW. $H \cup DW$ represents the question '*Do or do you not need a hammer?*', since this question can be answered by the statements $\{d \vee w, d, w, h, \bot\}$, where $d \vee w$ is equivalent to '*Fry does not need a hammer!*', since $\sim h = d \vee w$. [2] Note that not all of the questions in $\mathcal{Q}$ have English language equivalents.

4 THE QUESTION ALGEBRA

The ordered set $\mathcal{Q}$ is comprised of sets ordered by the usual set inclusion $\subseteq$. This ordering relation naturally implements the notion of *answering*. If a question A is defined by a set that is a subset of the answers to a second question B, so that $A \subseteq B$, then answering the question A will also answer the question B. Thus question A answers question B if and only if $A \subseteq B$. This allows us to read $A \subseteq B$ as 'A *answers* B', and recognize that questions lower in the lattice (Figure 2) answer questions higher in the lattice.

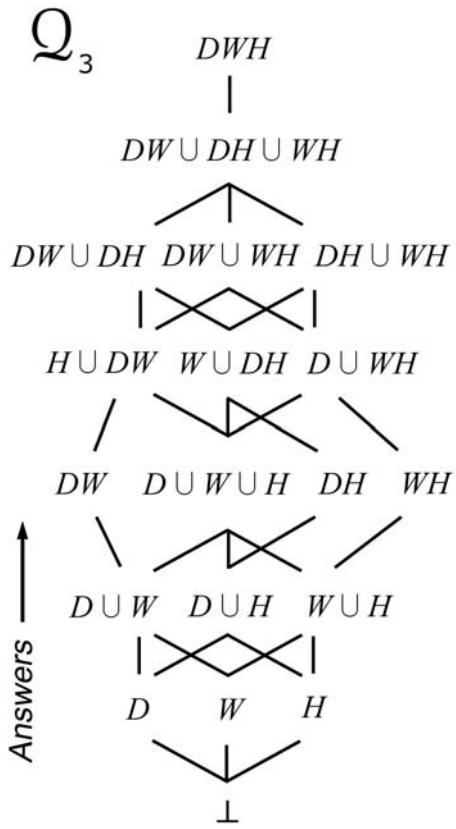


Figure 2: $\mathcal{Q}_3$ is the free distributive lattice formed from the assertion lattice $\mathcal{A}_3$. Questions are ordered by set-inclusion which implements the relation '*answers*'.

The fact that the ordered set $\mathcal{Q}$ is comprised of sets ordered by $\subseteq$ and closed under set union $\cup$ and set intersection $\cap$ implies that it is a *distributive lattice* (Knuth, 2003a,b, 2004a,b, 2005). This means that $\mathcal{Q}$ possesses two binary algebraic operations, the join $\vee$ and meet $\wedge$, which are identified with $\cup$ and $\cap$, respectively (Knuth, 2003a). Just as the join and meet on the assertion lattice $\mathcal{A}$ can be identified with the logical disjunction $\vee$ (OR) and the logical conjunction $\wedge$ (AND), the join and meet on the question lattice can also be viewed as a disjunction and a conjunction of questions, respectively. The question formed from the meet of two questions asks what the two questions ask jointly and is called the *joint question*; whereas the question formed from the join of two questions asks what the two questions ask in common and is called the *common question* (Cox, 1979). These operations

[2] Note also that $\bot$ is the absurd answer, which answers all questions since it implies everything (see Figure 1).

Table 1: The Question Algebra

ORDERING	
Answers	$\leq \equiv \subseteq$
Reflexivity	For all A, $A \leq A$
Antisymmetry	If $A \leq B$ and $B \leq A$ then $A = B$
Transitivity	If $A \leq B$ and $B \leq C$ then $A \leq C$
OPERATIONS	
Disjunction	$\vee \equiv \cup$
Conjunction	$\wedge \equiv \cap$
Idempotency	$A \vee A = A$
	$A \wedge A = A$
Commutativity	$A \vee B = B \vee A$
	$A \wedge B = B \wedge A$
Associativity	$A \vee (B \vee C) = (A \vee B) \vee C$
	$A \wedge (B \wedge C) = (A \wedge B) \wedge C$
Absorption	$A \vee (A \wedge B) = A \wedge (A \vee B) = A$
Distributivity	$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$
	$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$
CONSISTENCY	
$A \leq B \quad \Leftrightarrow \quad A \wedge B = A \quad \Leftrightarrow \quad A \vee B = B$	

allow us to algebraically manipulate questions as easily as we currently manipulate logical statements.

However, the similarities to the more specific Boolean algebra end there. Distributive algebras, in general, do not possess the Boolean operation of negation. Thus, in general, questions do not possess complements.

The join-irreducible elements of the question lattice $\mathcal{J}(\mathcal{Q})$ are the questions that cannot be written as the join (set union) of two other questions. I call these questions *ideal questions* (Knuth 2003a), denoted $\mathcal{I} = \mathcal{J}(\mathcal{Q})$, reflecting the fact that they are the *ideals* (Birkhoff, 1967; Davey & Priestley, 2002) of the lattice $\mathcal{Q}$. While ideal questions neither have a verbal analogue nor are interesting to ask, they are useful mathematical constructs. Ideal questions form a lattice isomorphic to the original assertion lattice $\mathcal{A}$. Thus we have the correspondence where $\mathcal{Q} = \mathcal{O}(\mathcal{A})$ and $\mathcal{A} \sim \mathcal{J}(\mathcal{Q})$. The lattices $\mathcal{A}$ and $\mathcal{Q}$ are said to be *dual* in the sense of Birkhoff's Representation Theorem (Knuth 2005). Furthermore, $\mathcal{O}$ takes lattice sums to lattice products; whereas $\mathcal{J}$ takes lattice products to lattice sums. These maps are the order-theoretic analogues of the exponential and the logarithm. This will have important consequences when we generalize the question algebra to the inquiry calculus.

There are other important types of questions. The first definition originated with Cox (1979).

Definition 3 (Real Question) *A real question is a question $Q \in \mathcal{Q}$, which can always be answered by a true statement. The real sublattice is denoted by $\mathcal{R}$.*

It is straightforward to show that a real question entertains each of the mutually exclusive atomic statements of $\mathcal{A}$ as acceptable answers (Knuth 2003a). This leads to the following proposition, which I will leave for the reader to prove.

Proposition 1 (The Least Real Question) *For all $Q \in \mathcal{Q}$, $Q \in \mathcal{R}$ iff $Q \geq \bigvee \downarrow\mathcal{J}(\mathcal{A})$. The question $C = \bigvee \downarrow\mathcal{J}(\mathcal{A}) = \min \mathcal{R}$ is the least real question.*

Thus the least element in the real sublattice $\mathcal{R}$ is the question formed from the join of the downsets of the mutually exclusive atomic statements of $\mathcal{A}$. In our example, this is $D \cup W \cup H$. This question is unique in that it answers all real questions in $\mathcal{Q}$.

Definition 4 (Central Issue) *The central issue is the least element in the real sublattice $\mathcal{R}$ of the question lattice $\mathcal{Q}$, denoted $\min \mathcal{R}$. Answering the central issue resolves all the real questions in the lattice.*

Last, a *partition question* is a real question that neatly partitions its set of answers. Specifically,

Definition 5 (Partition Question) *A partition question is a real question $P \in \mathcal{R}$ formed from the join of a set of ideal questions $P = \bigvee_{i=1}^{n} X_i$ where $\forall\, X_j, X_k \in \mathcal{J}(\mathcal{Q})$, $X_j \wedge X_k = \bot$ when $j \neq k$.*

There are five partition questions in our example: DWH, $H \cup DW$, $W \cup DH$, $D \cup WH$, and $D \cup W \cup H$. Together these questions form a lattice $\mathcal{P}$ isomorphic to the partition lattice $\mathbf{\Pi}_3$. Note that the central issue is the partition question with the maximal number of partitions. For this reason, it is the least ambiguous question.

The question lattice $\mathcal{Q}$ generated from the Boolean lattice $\mathcal{A}$ is known as the *free distributive lattice* (Knuth, 2003a). As such, it is isomorphic to the lattice of simplicial complexes in geometry (Klain & Rota, 1997), as well as the lattice of hypergraphs (Knuth, 2005). Thus

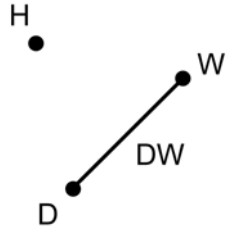


Figure 3: The hypergraph associated with the question $H \cup DW =$ '*Do you or do you not need a hammer?*'

hypergraphs are a convenient graphical means of diagramming questions. Figure 3 shows the hypergraph associated with the partition question $H \cup DW =$ '*Do you or do you not need a hammer?*' Such hypergraphs may play a more significant role when inquiry is united with inference in the form of Bayes Nets.

5 THE INQUIRY CALCULUS

With the question algebra well-defined, I now extend the ordering relation to a quantity that describes the *degree to which one question answers another*. This is done by defining a bi-valuation on the lattice that takes two questions and returns a real number $d \in [0, c]$, where c is the maximal relevance.[3] I call this bi-valuation the *relevance* (Knuth, 2005)

Definition 6 (Relevance) *The degree to which a question Q resolves an outstanding issue I, for all $Q, I \in \mathcal{Q}$, is called the relevance, and is written $d(I|Q)$ where*

$$d(I|Q) = \begin{cases} c & \text{if } Q \leq I \quad (Q \text{ answers } I) \\ 0 & \text{if } Q \wedge I = \bot \quad (Q \text{ and } I \text{ are exclusive}) \\ d & \text{otherwise, where } 0 < d < c. \end{cases}$$

with c being the maximal relevance.

This bi-valuation is defined so as to extend the dual of the zeta function for the lattice, which acts to quantify order-theoretic inclusion, which in this case, indicates whether the question Q answers the question I (Knuth, 2004a,b, 2005). The utility of this bi-valuation becomes apparent when one considers $I = \min \mathcal{R}$ to be the central issue, and $Q \in \mathcal{R}$ to be an arbitrary real question. The bi-valuation $d(I|Q)$ quantifies the degree to which Q resolves the central issue I by taking a value d where $0 < d < c$. This is analogous to the notation in probability theory where $p(x|y)$ describes the degree to which the statement y implies the statement x (Cox, 1946, 1961; Jaynes, 2003).

Since the arguments of the relevance function can be expressed as algebraic combinations of questions, we must require that the values returned by the function are consistent with the algebraic properties of the lattice. This consistency requirement results in three rules, which describe how relevances relate to one another (Knuth 2004a, 2005). Consistency with associativity gives rise to the ***Sum Rule*** (Caticha, 1998; Knuth 2004a, 2005):

$$d(X \vee Y|Q) = d(X|Q) + d(Y|Q) - d(X \wedge Y|Q), \quad (1)$$

and its multi-question generalization

$$d(X_1 \vee X_2 \vee \cdots \vee X_n|Q) = \sum_i d(X_i|Q) - \sum_{i<j} d(X_i \wedge X_j|Q) + \sum_{i<j<k} d(X_i \wedge X_j \wedge X_k|Q) - \cdots, \quad (2)$$

which, due to the Möbius function of the distributive lattice, displays the familiar sum and difference pattern known as the *inclusion-exclusion principle* (Klain & Rota, 1997; Knuth 2004a, 2005).

Consistency with distributivity of $\wedge$ over $\vee$ results in the ***Product Rule*** (Caticha, 1998; Knuth 2004a, 2005):

$$d(X \wedge Y|Q) = c\, d(X|Q) d(Y|X \wedge Q), \quad (3)$$

where the real number c is again the maximal relevance. Note that the calculus cannot simultaneously support distributivity of $\wedge$ over $\vee$ and distributivity of $\vee$ over $\wedge$, which are both allowed in a distributive lattice (Knuth 2004a, 2005). It may surprise some to learn that is also the case in probability theory.

Last, consistency with commutativity of $\wedge$ results in a ***Bayes' Theorem Analogue*** (Knuth 2004a, 2005):

$$d(Y|X \wedge Q) = \frac{d(Y|Q) d(X|Y \wedge Q)}{d(X|Q)}. \quad (4)$$

The fact that these three rules are shared between the inquiry calculus and probability theory is a result of the fact that both the assertion lattice $\mathcal{A}$ and the question lattice $\mathcal{Q}$ are distributive lattices, with a Boolean lattice being a special case of a distributive lattice.

Since the assertion lattice $\mathcal{A}$ and the question lattice $\mathcal{Q}$ are dual to one another in the sense of Birkhoff's Representation Theorem, it is not unreasonable to expect that the values of the relevances of questions must be consistent with the probabilities of their possible answers. Given an ideal question $X = \downarrow x$ we require

$$d(X|\top) = H(p(x|\top)), \quad (5)$$

where $d(X|\top)$ is the degree to which the question that asks everything $\top$ answers X, $p(x|\top)$ is the degree to which the truism (the top element of $\mathcal{A}$) implies the statement x, and H is a function to be determined.

The result, which I discuss in detail elsewhere (Knuth, 2005), is based on four constraints imposed by the lattice structure *additivity*, *subadditivity*, *symmetry*, and *expansibility*. Additivity and subadditivity are a result of the sum rule for questions (2). The constraint of symmetry reflects the commutativity of the join; whereas the constraint of expansibility reflects the fact

[3] Real numbers preserve transitivity, which is a useful property in this context. Are real numbers always appropriate in such generalizations? The answer is 'no' and quantum mechanics is an excellent example (Caticha, 1998; Knuth, 2004a).

that adding a statement that is known to be false to the underlying assertion lattice $\mathcal{A}$ does not affect the results. An important result from Janos Aczél and colleagues (Aczél, Forte & Ng, 1974) enables one to show that given these properties, there is a unique form of the relevance $d(P|\top)$ in terms of probabilities.

Theorem 1 (Relevance) *If and only if $d(P|\top)$ satisfies additivity, subadditivity, symmetry and expansibility, then there exist $a \geq 0$ and $b \geq 0$ such that*

$$d(P|\top) = a\ H_m(p_1, p_2, \cdots, p_n) + b\ {}_oH_m(p_1, p_2, \cdots, p_n), \quad (6)$$

where $p_i \equiv p(x_i|\top)$, the Shannon entropy (Shannon & Weaver, 1949) is defined as

$$H_m(p_1, p_2, \cdots, p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i, \quad (7)$$

and the Hartley entropy (Hartley, 1928) is defined as

$${}_oH_m(p_1, p_2, \cdots, p_n) = \log_2 N(P), \quad (8)$$

where $N(P)$ is the number of non-zero arguments p_i.

This result is important since it rules out the use of other entropies for the purpose of inference and inquiry. Any other entropy function will lead to an inconsistency between the bi-valuations defined on the assertion lattice $\mathcal{A}$ and the bi-valuations defined on the question lattice $\mathcal{Q}$.

6 A NATURAL GENERALIZATION OF INFORMATION THEORY

I will now show that these results not only lead naturally to information theory, but significantly generalize its scope including several generalizations already proposed in the literature. For simplicity, I will assign the arbitrary constants so that $a = 1$ and $b = 0$, and limit ourselves to the Shannon entropy. The main result of the previous section is that the degree to which the top question $\top$ answers any partition question $P \in \mathcal{P}$ is quantified by the entropy of its answers. Thus probability quantifies what we know, whereas entropy quantifies what we do not know.

However, more basic quantities also appear, and take on new fundamental importance. Since partition questions are joins of ideal questions, it is straightforward to show, using the sum rule, that the degree to which $\top$ answers an ideal question $X_i \in \mathcal{I}$ is given by the probability-weighted surprise

$$d(X_i|\top) = -p_i \log_2 p_i. \quad (9)$$

If we look at our earlier example, we can compute the degree to which $\top$ answers the question $DW \vee WH$. This is easily done using the sum rule, which gives

$$d(DW \vee WH|\top) = d(DW|\top) + d(WH|\top) - d(DW \wedge WH|\top). \quad (10)$$

Clearly this quantity is related to the *mutual information* between DW and WH, which when written in standard notation would look like

$$I(DW; WH) = H(DW) + H(WH) - H(DW, WH), \quad (11)$$

where $d(DW \wedge WH|\top)$ is related to the *joint entropy*. Thus mutual information is related to the disjunction of two issues, whereas the joint entropy is related to the conjunction of two issues. However, in this illustration is important to note that (10) is not exactly a mutual information since neither DW nor WH are partition questions, however with a larger hypothesis space it is trivial to construct the mutual information this way.

By considering the disjunction and conjunction of multiple issues, one can construct relevances that are higher-order mutual informations and higher-order joint entropies that exhibit the sum and difference patterns in the multi-question generalization of the sum rule. Higher-order generalizations such as these were independently suggested by several authors (McGill, 1955; Cox, 1961, 1979; Bell, 2003), however here one can see that they occur naturally as a result of the inquiry calculus.

To consider the conjunction and disjunction of questions to the right of the solidus, one must use the sum and product rules in conjunction with the Bayes' theorem analogue to move questions from one side of the solidus to the other. The following example demonstrates a typical calculation, which also includes some algebraic manipulation. Consider again Bender's central issue $T =$ '*Which tool do you need?*'. However, Bender has asked this question 10 times in the last hour, and Fry is getting quite irritated and will lose his temper if he hears that question again. To find another question, Bender computes the relevance that the question $Q_H =$ '*Do you or do you not need a hammer?*' has on the issue. This calculation results in

$$\begin{aligned} d(T|Q_H) &= d(T|Q_H \wedge \top) \quad (12)\\ &= d(Q_H|T \wedge \top)\frac{d(Q_H|\top)}{d(T|\top)}\\ &= d(Q_H|T)\frac{d(Q_H|\top)}{d(T|\top)}\\ &= c\frac{d(Q_H|\top)}{d(T|\top)}, \end{aligned}$$

where the result is simply a ratio of two entropies. Note that this formalism relies on relevances that are conditional—like probabilities. This notion is absent in traditional information theory, and is another way in which the inquiry calculus is a natural generalization.

7 DISCUSSION

I have demonstrated that the question algebra and the inquiry calculus follow naturally from a straightforward definition of a question as the set of statements that answer it. The question algebra enables one to manipulate questions algebraically as easily as we currently manipulate logical statements, whereas the inquiry calculus allows us to quantify the degree to which one question answers another. This methodology promises to enable us to design machines that can identify maximally relevant questions in order to actively obtain information. This work has clear implications for areas of research that rely on question-asking, such as experimental design (Lindley, 1956; Loredo, 2004), search theory (Pierce, 1979), and active learning (MacKay, 1992), each of which has taken advantage of information theory during their histories. In addition, this approach has already shown promise in several applications by Robert Fry (1995, 2002).

However, the inquiry calculus is more fundamental than information theory in the sense that it derives directly from the question algebra. The sole postulate is that the bi-valuations on the dual lattices are defined consistently. The result is that the Shannon and Hartley entropies are the only entropies that can be used for the purposes of inquiry—all other entropies will lead to inconsistencies. Entropy is related to the relevances involving the partition questions, mutual information is related to disjunctions of questions, and joint entropy is related to conjunctions of questions. Higher-order informations occur naturally when multiple disjunctions and conjunctions are considered. Last, the calculus allows for, and relies on, conditional quantities not considered in traditional information theory. The result is an algebra and a calculus that takes the guesswork out of defining information-theoretic cost functions in applications involving question-asking.

Our explorations into the realm of questions are only beginning, and it would be naïve to think that the work presented here is the entire story. Recently, Ariel Caticha presented an alternative approach to viewing a question as a probability distribution, which is in some ways simultaneously more general yet more restrictive than the approach presented here (Caticha, 2004). The result is a measure of relevance described by relative entropy. It will be interesting to see where these new investigations lead.

APPENDIX: POSETS AND LATTICES

In this section I introduce some basic concepts of order theory that are necessary to understand the spaces of logical statements and questions. Order theory captures the notion of ordering elements of a set. For a given set, one associates a *binary ordering relation* to form what is called a *partially-ordered set*, or a *poset* for short. This ordering relation, generically written $\leq$, satisfies reflexivity, antisymmetry, and transitivity. The ordering $a \leq b$ is generally read 'b includes a'. When $a \leq b$ and $a \neq b$, we write $a < b$. Furthermore, if $a < b$, but there does not exist an element x in the set such that $a < x < b$, then we write $a \prec b$, read 'b covers a', indicating that b is a direct successor to a in the hierarchy induced by the ordering relation. This concept of covering can be used to construct diagrams of a poset. If an element b includes an element a then it is drawn higher in the diagram. If b covers a then they are connected by a line.

A poset P possesses a greatest element if there exists an element $\top \in P$, called the *top*, where $x \leq \top$ for all $x \in P$. Dually, a poset may possess a least element $\bot \in P$, called the *bottom*. The elements that cover the bottom are called *atoms*.

Given two elements x and y, their *upper bound* is defined as the set of all $z \in P$ such that $x \leq z$ and $y \leq z$. If a unique *least upper bound* exists, it is called the *join*, written $x \vee y$. Dually, we can define the *lower bound* and the *greatest lower bound*, which if it exists, is called the *meet*, $x \wedge y$. Graphically the join of two elements can be found by following the lines upward until they first converge on a single element. The meet can be found dually. Elements that cannot be expressed as a join of two elements belong to a special set of elements called *join-irreducible elements*.

The dual of a poset P, written P^{∂} can be formed by reversing the ordering relation, which can be visualized by flipping the poset diagram upside-down. This action exchanges joins and meets and is the reason that their relations come in pairs (see Table 1).

A *lattice* $\mathfrak{L}$ is a poset where the join and meet exist for every pair of elements. We can view the lattice from a structural viewpoint as a set of objects arranged by an ordering relation $\leq$. However, we can also view the lattice from an operational viewpoint as an algebra on the space of elements with the operations $\vee$ and $\wedge$ along with any other relations induced by the ordering relation. The join and meet obey idempotency, commutativity, associativity, and the absorption property.

Acknowledgements

This work was supported by the NASA IDU/IS/CICT Program and the NASA Aerospace Technology Enterprise. I am deeply indebted to Ariel Caticha, Bob Fry, Janos Aczél and Kevin Wheeler for insightful and inspiring discussions, and the anonymous reviewers for their detailed and helpful comments.

References

Aczél J., Forte B. & Ng C.T. (1974). Why the Shannon and Hartley entropies are 'natural'. *Adv. Appl. Prob.*, Vol. 6, pp. 131–146.

Bell A.J. (2003). The co-information lattice. *Proceedings of the Fifth International Workshop on Independent Component Analysis and Blind Signal Separation: ICA 2003* (eds. S. Amari, A. Cichocki, S. Makino and N. Murata).

Birkhoff G.D. (1967). *Lattice Theory*, Providence:American Mathematical Society.

Boole G. (1854). *An Investigation of the Laws of Thought.* London:Macmillan.

Caticha A. (1998). Consistency, amplitudes and probabilities in quantum theory. *Phys. Rev. A*, Vol. 57, pp. 1572–1582.

Caticha A. (2004). Questions, relevance and relative entropy. In press: *Bayesian Inference and Maximum Entropy Methods in Science and Engineering, Garching, Germany, August 2004* (eds. R. Fischer, R. Preuss, U. Von Toussaint, V. Dose). AIP Conf. Proc., Melville NY:AIP.

Cox R.T. (1946). Probability, frequency, and reasonable expectation. *Am. J. Physics*, Vol. 14, pp. 1–13.

Cox R.T. (1961). *The algebra of probable inference.* Baltimore:Johns Hopkins Press.

Cox R.T. (1979). Of inference and inquiry. In *The Maximum Entropy Formalism* (eds. R. D. Levine & M. Tribus). Cambridge:MIT Press, pp. 119–167.

Davey B.A. & Priestley H.A. (2002). *Introduction to Lattices and Order.* Cambridge:Cambridge Univ. Press.

Fry R.L. (1995). Observer-participant models of neural processing. *IEEE Trans. Neural Networks*, Vol. 6, pp. 918–928.

Fry R.L. (2002). The engineering of cybernetic systems. In *Bayesian Inference and Maximum Entropy Methods in Science and Engineering, Baltimore MD, USA, August 2001* (ed. R. L. Fry). New York:AIP, pp. 497–528.

Hartley R.V. (1928). Transmission of information. *Bell System Tech. J.*, Vol. 7, pp. 535–563.

Jaynes E.T. (2003). *Probability theory: the logic of science.* Cambridge:Cambridge Univ. Press.

Klain D.A. & Rota G.-C. (1997). *Introduction to geometric probability.* Cambridge:Cambridge Univ. Press.

Knuth K.H. (2003a). What is a question? In *Bayesian Inference and Maximum Entropy Methods in Science and Engineering, Moscow ID, USA, August 2002* (ed. C. Williams). AIP Conf. Proc. Vol. 659, Melville NY:AIP, pp. 227–242.

Knuth K.H. (2003b). Intelligent machines in the 21st century: foundations of inference and inquiry, *Phil. Trans. Roy. Soc. Lond. A*, Vol. 361, No. 1813, pp. 2859–2873.

Knuth K.H. (2004a). Deriving laws from ordering relations. In *Bayesian Inference and Maximum Entropy Methods in Science and Engineering, Jackson Hole WY, USA, August 2003* (ed. G. J. Erickson). AIP Conf. Proc. Vol. 707, Melville NY:AIP, pp. 204–235.

Knuth K.H. (2004b). Measuring questions: Relevance and its relation to entropy. In press: *Bayesian Inference and Maximum Entropy Methods in Science and Engineering, Garching, Germany, August 2004* (eds. R. Fischer, R. Preuss, U. Von Toussaint, V. Dose). AIP Conf. Proc., Melville NY:AIP.

Knuth K.H. (2005). Lattice duality: The origin of probability and entropy. In press: *Neurocomputing.*

Lindley D.V. (1956). On the measure of information provided by an experiment. *Ann. Math. Statist.* Vol. 27, pp. 986–1005.

Loredo T.J. (2004). Bayesian adaptive exploration. In: *Bayesian Inference and Maximum Entropy Methods in Science and Engineering, Jackson Hole WY, USA, August 2003* (ed. G. J. Erickson). AIP Conf. Proc. Vol. 707, Melville NY:AIP, pp. 330–346.

MacKay D.J.C. (1992). Information-based objective functions for active data selection. *Neural Computation* Vol. 4 No. 4, pp. 589–603.

McGill W.J. (1955). Multivariate information transmission. *IEEE Trans Info Theory*, Vol. 4, pp. 93–111.

Pierce J.G. (1979). A new look at the relation between information theory and search theory. In *The Maximum Entropy Formalism* (eds. R. D. Levine & M. Tribus), Cambridge:MIT Press, pp. 339–402.

Shannon C.E. & Weaver W. (1949). *A mathematical theory of communication.* Chicago:Univ. of Ill. Press.

Convergent tree-reweighted message passing for energy minimization

Vladimir Kolmogorov
Microsoft Research
Cambridge, UK
vnk@microsoft.com

Abstract

Tree-reweighted max-product message passing (TRW) is an algorithm for energy minimization introduced recently by Wainwright et al. [7]. It shares some similarities with Pearl's loopy belief propagation. TRW was inspired by a problem of maximizing a lower bound on the energy. However, the algorithm is not guaranteed to increase this bound - it may actually go down. In addition, TRW does not always converge. We develop a modification of this algorithm which we call *sequential tree-reweighted message passing*. Its main property is that the bound is guaranteed not to decrease. We also give a *weak tree agreement* condition which characterizes local maxima of the bound with respect to TRW algorithms. We prove that our algorithm has a limit point that achieves weak tree agreement. Experimental results demonstrate that on certain synthetic and real problems our algorithm outperforms both the ordinary belief propagation and tree-reweighted algorithm [7].

1 Introduction

Sum-product algorithms Pearl's loopy belief propagation ("sum-product BP") is a popular algorithm for inference in Bayesian networks. If the network is a tree, then it converges in a finite number of iterations, and the solution gives exact marginals. However, if the network contains loops then convergence is not guaranteed.

Fixed points of BP have been shown to correspond to extrema of the so-called Bethe free energy [12]. This motivated algorithms for direct minimization of the Bethe free energy, such as CCCP [13] and UPS [5]. They are guaranteed to converge; however, the computation cost is often higher than the cost of BP.

Several researchers proposed alternatives to the Bethe free energy [6, 9, 11, 3]. Functionals used in [6, 9] are convex upper bounds on the log partition function, and functionals in [3] are convex upper bounds on the Bethe free energy. In order to minimize these functionals, belief propagation algorithm was modified in such a way that its fixed points correspond to extrema of the functional.

The algorithm proposed in [9] is called *tree-reweighted message passing* (TRW). Interestingly, its fixed point achieves the *global* minimum of the upper bound. Unfortunately, as in the case of ordinary BP, convergence is not guaranteed.

Max-product algorithms Closely related to sum-product are "max-product" (or "min-sum") BP algorithms. Their goal is to find a configuration with the maximum a posteriori (MAP) probability, or a configuration with the smallest energy. Generally, max-product algorithms can be obtained from sum-product versions in the zero temperature limit. There are caveats, however; for example, it is not known whether fixed points of max-product BP correspond to extrema of some functional.

In this paper we focus on the max-product version of tree-reweighted algorithm [7]. In fact, two different TRW algorithms are developed in [7]. They are inspired by the problem of maximizing a concave lower bound on the energy. These algorithms have the following property: if their fixed point satisfies a certain condition ("tree agreement") then it is guaranteed to give a MAP solution (i.e. a global minimum of the energy).

However, TRW algorithms in [7] cannot be viewed as algorithms for direct maximization of the bound. Indeed, in our experiments we observed that sometimes they decrease it. Also, the algorithms do not always converge; when it happens, the value of the bound of-

ten goes into a loop.

Our main contribution is as follows: we show how to modify TRW algorithms so that the value of the bound is guaranteed not to decrease. Thus, we are guaranteed to find at least a "local" maximum of the bound. The word "local" is in quotes since for concave functions all local maxima are global, if the standard metric space topology is used. Here we use a weaker topology: our maxima are local with respect to the TRW algorithms. We formulate the *weak tree agreement* condition (WTA) which gives a precise characterization of such maxima. We prove that our algorithm has a subsequence converging to a vector satisfying WTA.

An interesting question is whether WTA always gives a global maximum of the bound. We show that this is not the case by providing a counterexample. This is a difference between sum-product and max-product cases.

TRW algorithms require some choice of trees covering the graph. If the trees have a special structure (namely, chains which are *monotonic* with respect to some ordering on the graph) then our algorithm reduces to the TRW message-passing algorithm of Wainwright et al. [7], but with a significant distinction: we update messages in a specific *sequential* order rather than in parallel. In the context of ordinary BP it is a well-known experimental fact that sequential updates are superior to parallel updates, although convergence is still not guaranteed. We give a theoretical justification of sequential updates in the case of tree-reweighted algorithms.

Our experimental results include both synthetic and real problems. In particular, we consider an energy function arising in the stereo matching problem [1]. We demonstrate that our algorithm outperforms both the ordinary BP and the TRW algorithms of Wainwright et al. [7]. Moreover, we obtain a slightly lower energy than the expansion move method [1], which is generally considered to be the most accurate minimization technique for such energy functions.

Outline The paper is organized as follows. In section 2 we introduce our notation and review some results from [7], in particular the lower bound on the energy function via convex combination of trees and duality result. Our new tree-reweighted algorithm and its analysis are given in section 3. Experimental results are described in section 4. Finally, we give conclusions in section 5.

2 Notation and background

In this paper we closely follow the notation used in [7]. However, instead of maximizing posterior probability we minimize an energy function. Therefore, we replace "min" with "max", "inf" with "sup" and vice versa.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with the set of vertices $\mathcal{V}$ and the set of edges $\mathcal{E}$. For each $s \in \mathcal{V}$, let x_s be a variable taking values in some discrete space $\mathcal{X}_s$. By concatenating the variables at each node, we obtain a vector $\mathbf{x}$ with $n = |\mathcal{V}|$ elements. This vector takes values in the space $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_n$. Unless noted otherwise, symbols s and t will denote nodes in $\mathcal{V}$, (s,t) - edge in $\mathcal{E}$, j and k - variables in $\mathcal{X}_s$ and $\mathcal{X}_t$, respectively.

A *potential function* is a mapping $\phi : \mathcal{X} \to \mathbb{R}$. We can also consider a family of potential functions $\{\phi_\alpha \mid \alpha \in \mathcal{I}\}$ where $\mathcal{I}$ is some index set. This family defines a vector-valued mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^d$. Associated with $\boldsymbol{\phi}$ is a real-valued vector $\theta = \{\theta_\alpha \mid \alpha \in \mathcal{I}\}$, which we call an *energy parameter vector*[1]. This vector defines an energy function $E(\cdot \mid \theta) : \mathcal{X} \to \mathbb{R}$ as follows:

$$E(\mathbf{x} \mid \theta) = \langle \theta, \boldsymbol{\phi}(\mathbf{x}) \rangle = \sum_{\alpha \in \mathcal{I}} \theta_\alpha \phi_\alpha(\mathbf{x}) \qquad (1)$$

where $\langle \cdot, \cdot \rangle$ is the ordinary Euclidean product in $\mathbb{R}^d$.

We will use the collection of potential functions called the *canonical overcomplete representation* [8, 10]. It is defined as follows:

$$\{\delta_j(x_s)\} \cup \{\delta_j(x_s)\delta_k(x_t)\} \cup \{\phi_{const}(\mathbf{x})\}$$

where $\delta_j(x_s)$ is an *indicator function* - it is equal to one if $x_s = j$, and zero otherwise. Function $\phi_{const}(\mathbf{x})$ equals to 1 for all arguments. The index set for this representation is

$$\mathcal{I} = \{(s;j)\} \cup \{(st;jk)\} \cup \{const\}$$

Note that $(st;jk) \equiv (ts;kj)$, so $\theta_{st;jk}$ and $\theta_{ts;kj}$ are the same element.

Sometimes it will be convenient to denote elements $\theta_{s;j}$ and $\theta_{st;jk}$ as $\theta_s(j)$ and $\theta_{st}(j,k)$, respectively. We will also use notation θ_s to denote a vector of size $|\mathcal{X}_s|$ and θ_{st} to denote a vector of size $|\mathcal{X}_s \times \mathcal{X}_t|$.

Note that energy function 1 corresponding to this representation can be written as a sum of unary terms corresponding to nodes in $\mathcal{V}$, and pairwise terms corresponding to edges in $\mathcal{E}$:

$$E(\mathbf{x} \mid \theta) = \theta_{const} + \sum_{s \in \mathcal{V}} \theta_s(x_s) + \sum_{(s,t) \in \mathcal{E}} \theta_{st}(x_s, x_t)$$

Let us introduce another notation which we will use extensively throughout the paper. Functions $\Phi, \Phi_{s;j}, \Phi_{st;jk} : \mathbb{R}^d \to \mathbb{R}$ give information about the

[1] In [8, 10] parameter θ is called an *exponential parameter vector*. We use a different name to emphasize the fact that our θ is the negative of θ used in [8, 10].

minimum values of the energy under different constraints:

$$\begin{array}{lll} \Phi(\theta) & = \min_{\mathbf{x}\in\mathcal{X}} & E(\mathbf{x} \mid \theta) \\ \Phi_{s;j}(\theta) & = \min_{\mathbf{x}\in\mathcal{X}, x_s=j} & E(\mathbf{x} \mid \theta) \\ \Phi_{st;jk}(\theta) & = \min_{\mathbf{x}\in\mathcal{X}, x_s=j, x_t=k} & E(\mathbf{x} \mid \theta) \end{array}$$

Values $\Phi_{s;j}(\theta)$ and $\Phi_{st;jk}(\theta)$ are called *min-marginals* for node s and edge (s,t), respectively.

2.1 Reparameterization and max-product belief propagation

If two parameter vectors θ and θ' define the same energy function (i.e. $E(\mathbf{x} \mid \theta') = E(\mathbf{x} \mid \theta)$ for all $\mathbf{x} \in \mathcal{X}$) then θ' is called a *reparameterization* of θ [8, 10]. We will write this as $\theta' \equiv \theta$. Note that this condition does not necessarily imply that $\theta' = \theta$ since there are various linear relations among potential functions ϕ_α.

Reparameterization provides an alternative tool for the analysis of belief propagation algorithm. Recall that a basic operation of BP is *passing a message* from node s to node t. As shown in [8, 10], this operation can be implemented without any messages: it is equivalent to a certain reparameterization of vectors θ_{st} and θ_t for edge (s,t) and node t, respectively. We will say that θ is in a *normal form* if it is a fixed point of BP.

If graph $\mathcal{G}$ is a tree, then values $\theta_{s;j}$ and $\theta_{st;jk}$ for vector θ in a normal form have a particularly simple interpretation [10] - they correspond to min-marginals (up to a constant):

$$\begin{array}{lll} \Phi_{s;j}(\theta) & = & \theta_{s;j} + const_s \\ \Phi_{st;jk}(\theta) & = & \theta_{s;j} + \theta_{st;jk} + \theta_{t;k} + const_{st} \end{array} \quad (2)$$

where $const_s$ and $const_{st}$ are constants independent of j and k.

2.2 Lower bound on the energy function

In this section we review the bound proposed in [7]. In order to describe it, we need to introduce some notation.

Let $\mathcal{T}$ be a collection of trees in graph $\mathcal{G}$ and $\rho^T,\ T \in \mathcal{T}$ be some distribution on $\mathcal{T}$. Throughout the paper we assume that each tree has a non-zero probability and each edge in $\mathcal{E}$ is covered by at least one tree.

For a given tree $T = (\mathcal{V}^T, \mathcal{E}^T)$ we define a set

$$\mathcal{I}^T = \{(s;j) \mid s \in \mathcal{V}^T\} \cup \{(st;jk) \mid (s,t) \in \mathcal{E}^T\} \cup \{const\}$$

corresponding to those indexes associated with vertices and edges in the tree.

To each tree $T \in \mathcal{T}$, we associate an energy parameter θ^T that must respect the structure of T. More precisely, the parameter θ^T must belong to the following linear constraint set:

$$\mathcal{A}^T = \{\theta^T \in \mathbb{R}^d \mid \theta^T_\alpha = 0 \ \ \forall\, \alpha \in \mathcal{I} \backslash \mathcal{I}^T\}$$

By concatenating all of the tree vectors, we form a larger vector $\boldsymbol{\theta} = \{\theta^T \mid T \in \mathcal{T}\}$, which is an element of $\mathbb{R}^{d \times |\mathcal{T}|}$. Vector $\boldsymbol{\theta}$ must belong to the constraint set

$$\mathcal{A} = \{\boldsymbol{\theta} \in \mathbb{R}^{d \times |\mathcal{T}|} \mid \theta^T \in \mathcal{A}^T \text{ for all } T \in \mathcal{T}\}$$

Consider function $\Phi_\rho : \mathcal{A} \to \mathbb{R}$ defined as follows:

$$\Phi_\rho(\boldsymbol{\theta}) = \sum_T \rho^T \Phi(\theta^T) = \sum_T \rho^T \min_{\mathbf{x}\in\mathcal{X}} \langle \theta^T, \boldsymbol{\phi}(\mathbf{x}) \rangle$$

[7] shows that if $\sum_T \rho^T \theta^T = \bar{\theta}$ then $\Phi_\rho(\boldsymbol{\theta})$ is a lower bound on the optimal value of the energy for vector $\bar{\theta}$ (this follows from Jensen's inequality). To get the tightest bound we can consider the following maximization problem:

$$\max_{\boldsymbol{\theta}\in\mathcal{A}, \sum_T \rho^T \theta^T = \bar{\theta}} \Phi_\rho(\boldsymbol{\theta}) \quad (3)$$

Interestingly, the optimal value of this problem does not depend on the choice of trees [7]. A proof of this fact can be summarized as follows. Φ_ρ is a concave function of $\boldsymbol{\theta}$; moreover, the constraints of problem 3 are linear in $\boldsymbol{\theta}$. Thus, we can consider its Lagrangian dual. This dual problem turns out to be a certain linear programming relaxation of the original minimization problem, and does not involve any trees[2].

3 New tree-reweighted message passing algorithm

Maximization problem 3 inspired two algorithms in [7] - tree-reweighted message passing with edge based updates (TRW-E) and with tree based updates (TRW-T). However, neither algorithm maintains constraint $\sum_T \rho^T \theta^T = \bar{\theta}$ of problem 3. Indeed, they perform reparameterizations of the original parameter vector, so this equality may become violated[3]. Let us replace it with the constraint $\sum_T \rho^T \theta^T \equiv \bar{\theta}$. Thus, we are now interested in the following maximization problem:

$$\max_{\boldsymbol{\theta}\in\mathcal{A}, \sum_T \rho^T \theta^T \equiv \bar{\theta}} \Phi_\rho(\boldsymbol{\theta}) \quad (4)$$

The following lemma justifies this formulation.

Lemma 3.1. *The optimal value of problem 4 equals to the optimal value of problem 3.*

[2] [7] formulated the duality theorem for the case when trees in $\mathcal{T}$ are *spanning*. However, their proof never uses this assumption. In this paper we do not assume that trees are spanning.

[3] Note that lemmas 5 and 11 in [7] seem to contain a mistake. Lemma 11, for example, says that TRW-T algorithm maintains the property $\sum_T \rho^T \theta^T = \bar{\theta}$. However, the proof does not take into account reparameterization step.

Proof. See [4]. The proof involves showing that any reparameterization can be expressed via messages. It is omitted due to space limitations. □

As shown in [7], TRW-E and TRW-T algorithms maintain the constraint of problem 4. Unfortunately, they do not guarantee that the objective function Φ_ρ monotonically increases - in our experiments we have observed that sometimes it goes down. In fact, when the algorithms failed to converge the value of $\Phi_\rho(\boldsymbol{\theta})$ often had gone into a loop. Next we design a new algorithm with the property that Φ_ρ never decreases.

Our algorithm is shown in Fig. 1. Unlike TRW-E and TRW-T algorithms which use parallel updates, we update vectors $\{\theta^T\}$ sequentially. Therefore, we call our algorithm "sequential tree-reweighted message passing" (TRW-S).

Reparameterization step 1(a) can be implemented in many different ways. One possibility is to convert vectors θ^T to normal forms my running the ordinary max-product BP[4]. However, this would be very expensive if trees are large. A more efficient technique is discussed in section 3.3.

3.1 Weak tree agreement

The algorithm in Fig. 1 does not specify what the stopping criterion is. In this section we address this issue by giving *weak tree agreement* condition (WTA). Later we will show that it characterizes local maxima of the algorithm with respect to function Φ_ρ. More precisely, we will prove that the algorithm has a subsequence converging to a vector satisfying WTA condition. Moreover, if a vector satisfies these conditions, then the algorithm will not make any progress, i.e. it will not increase function Φ_ρ.

In order to define this condition, it is convenient to introduce some notation. Let $\mathrm{OPT}^T(\theta^T)$ be the set of optimal configurations for parameter θ^T. Let $\mathrm{OPT}(\boldsymbol{\theta})$ be the collection $\{\mathrm{OPT}^T(\theta^T) \mid T \in \mathcal{T}\}$ of the sets of optimal configurations for vectors θ^T. It belongs to the set $(2^{\mathcal{X}})^{|\mathcal{T}|} = 2^{\mathcal{X}} \times \ldots \times 2^{\mathcal{X}}$ ($|\mathcal{T}|$ times). For two collections $\mathbb{S}, \tilde{\mathbb{S}} \in (2^{\mathcal{X}})^{|\mathcal{T}|}$ we will write $\mathbb{S} \subset \tilde{\mathbb{S}}$ if $\mathbb{S}^T \subset \tilde{\mathbb{S}}^T$ for any tree T.

Consider some collection of sets of configurations $\mathbb{S} = \{\mathbb{S}^T\} \in (2^{\mathcal{X}})^{|\mathcal{T}|}$. We say that $\mathbb{S}$ is *consistent* if it satisfies the following three conditions:

(a) For any tree T set $\mathbb{S}^T$ is non-empty.

[4]With this scheme a connection to TRW-T algorithm is more apparent. Basically, TRW-T iterates between two phases: (a) running max-product BP for *all* trees, and (b) performing averaging operation for *all* nodes and edges.

0. Initialize $\boldsymbol{\theta}$ so that $\boldsymbol{\theta} \in \mathcal{A}$ and $\sum_T \rho^T \theta^T \equiv \bar{\theta}$.

1. Select some order for nodes and edges in $\mathcal{V} \cup \mathcal{E}$. For each element $\omega \in \mathcal{V} \cup \mathcal{E}$ find all trees $\mathcal{T}_\omega \subset \mathcal{T}$ containing ω. If there is more than one tree, then do the following:

(a) For all trees $T \in \mathcal{T}_\omega$ reparameterize θ^T such that values $\theta^T_{s;j}$ (if $\omega = s$ is a node) or $\theta^T_{s;j} + \theta^T_{st;jk} + \theta^T_{t;k}$ (if $\omega = (s,t)$ is an edge) give correct min-marginals for tree T as in formula 2.

(b) "Averaging" operation:
If $\omega = s$ is a node in $\mathcal{V}$ then
- Compute $\tilde{\theta}_s = \frac{1}{\rho_s} \sum_{T \in \mathcal{T}_s} \rho^T \theta^T_s$
- Set $\theta^T_s := \tilde{\theta}_s$ for trees $T \in \mathcal{T}_s$

If $\omega = (s,t)$ is an edge in $\mathcal{E}$ then
- Compute

$$\tilde{\nu}_{st;jk} = \frac{1}{\rho_{st}} \sum_{T \in \mathcal{T}_{st}} \rho^T (\theta^T_{s;j} + \theta^T_{st;jk} + \theta^T_{t;k})$$

- Set θ^T_s, θ^T_{st}, θ^T_t for trees $T \in \mathcal{T}_{st}$ so that

$$\theta^T_{s;j} + \theta^T_{st;jk} + \theta^T_{t;k} = \tilde{\nu}_{st;jk}$$

2. Check whether a stopping criterion is satisfied; if yes, terminate, otherwise go to step 1.

Figure 1: **Sequential tree-reweighted algorithm (TRW-S).**

(b) If node s is contained in trees T and T', then for any configuration $\mathbf{x}^T \in \mathbb{S}^T$ there exists configuration $\mathbf{x}^{T'} \in \mathbb{S}^{T'}$ which agrees with $\mathbf{x}^T$ on node s, i.e. $x^T_s = x^{T'}_s$.

(c) If edge (s,t) is contained in trees T and T', then for any configuration $\mathbf{x}^T \in \mathbb{S}^T$ there exists configuration $\mathbf{x}^{T'} \in \mathbb{S}^{T'}$ which agrees with $\mathbf{x}^T$ on nodes s and t, i.e. $x^T_s = x^{T'}_s$, $x^T_t = x^{T'}_t$.

Now we can define WTA condition.

Definition 3.2. *Vector $\boldsymbol{\theta} = \{\theta^T\} \in \mathcal{A}$ is said to satisfy the* weak tree agreement *condition if there exists collection $\mathbb{S} \subset OPT(\boldsymbol{\theta})$ which is consistent.*

Note that it can be viewed as a generalization of the *tree agreement* condition introduced in [7]: vectors satisfying tree agreement also satisfy WTA condition.

Also note that WTA condition is different from the *fixed point* condition of TRW algorithms. The latter means that any step of the algorithm does not change vector $\boldsymbol{\theta}$. This in turn implies that all vectors θ^T are in a normal form and $\theta^T_\omega = \theta^{T'}_\omega$ for every element $\omega \in \mathcal{V} \cup \mathcal{E}$ and for every pair of trees $T, T' \in \mathcal{T}_\omega$. It is easy

to see that every fixed point of TRW satisfies WTA condition, but not the other way around.

3.2 Analysis of the algorithm

First we show that similarly to TRW-T algorithm, TRW-S maintains the constraint of problem 4.

Lemma 3.3. *TRW-S algorithm performs reparameterization of the original parameter vector $\bar{\theta}$, i.e. it maintains the property $\sum_T \rho^T \theta^T \equiv \bar{\theta}$.*

This lemma follows directly from the algorithm's construction; see [4] for details.

Next we analyze the behaviour of objective function Φ_ρ during the algorithm. To be specific, we assume for the rest of this section that after reparameterization step 1(a) we have $\min_j \{\theta^T_{s;j}\} = 0$ (if $\omega = s$ is a node) or $\min_{j,k} \{\theta^T_{s;j} + \theta^T_{st;jk} + \theta^T_{t;k}\} = 0$ (if $\omega = (s,t)$ is an edge). This assumption is not essential, however; the behaviour of the algorithm will be the same for any other normalization.

Theorem 3.4. *(a) After any number of steps functions Φ and Φ_ρ do not decrease.*

(b) If vector $\boldsymbol{\theta}$ does not satisfy WTA condition then after a finite number of steps Φ_ρ will increase.

(c) If vector $\boldsymbol{\theta}$ satisfies WTA condition with collection $\mathbb{S}$ then after any number of steps it will still satisfy WTA with the same collection $\mathbb{S}$. Function Φ_ρ will not change.

Proof. Due to space limitations we prove only part (a) for the case of averaging operation for node s. The rest of the proof is in [4].

Suppose that this operation transforms vectors θ^T to vectors $\tilde{\theta}^T$. We need to show that $\Phi(\tilde{\theta}^T) \geq \Phi(\theta^T)$ for any tree $T \in \mathcal{T}_s$. This will imply $\sum_T \rho^T \Phi(\tilde{\theta}^T) \geq \sum_T \rho^T \Phi(\theta^T)$.

Because of our normalization assumption we have

$$\Phi(\theta^T) = \min_{j \in \mathcal{X}_s} \Phi_{s;j}(\theta^T) = \min_{j \in \mathcal{X}_s} \{\theta^T_{s;j}\} + const_s = const_s$$

where $const_s$ is the constant in formula 2. Plugging this into formula 2 we get $\Phi_{s;j}(\theta^T) = \Phi(\theta^T) + \theta^T_{s;j}$.

Vectors θ^T and $\tilde{\theta}^T$ agree on all nodes and edges other than node s. Thus, they have the same optimal configuration $\mathbf{x}^j$ with the constraint $x^j_s = j$. We can write

$$\Phi_{s;j}(\tilde{\theta}^T) = E(\mathbf{x}^j \mid \tilde{\theta}^T) = E(\mathbf{x}^j \mid \theta^T) - \theta^T_{s;j} + \tilde{\theta}^T_{s;j} =$$

$$= \Phi_{s;j}(\theta^T) - \theta^T_{s;j} + \tilde{\theta}^T_{s;j} = \Phi(\theta^T) + \tilde{\theta}^T_{s;j}$$

$$\Phi(\tilde{\theta}^T) = \min_{j \in \mathcal{X}_s} \Phi_{s;j}(\tilde{\theta}^T) = \Phi(\theta^T) + \min_{j \in \mathcal{X}_s} \{\tilde{\theta}^T_{s;j}\}$$

We have $\theta^T_s \geq 0$; inspecting update rule of step 1(b) we conclude that $\tilde{\theta}^T_s \geq 0$ as well. Therefore, the minimum on the RHS is non-negative, so $\Phi(\tilde{\theta}^T) \geq \Phi(\theta^T)$. □

As an immediate consequence of theorem 3.4(b) we get the following result:

Corollary 3.5. *If vector $\boldsymbol{\theta}$ maximizes problem 4 then $\boldsymbol{\theta}$ satisfies WTA condition.*

Unfortunately, the converse is not necessarily true as example in [4] demonstrates.

Finally, we give the convergence theorem. A proof is given in [4]; here we omit it due to space limitations.

Theorem 3.6. *Let $\{\boldsymbol{\theta}^{(i)}\}_i$ be an infinite sequence of vectors obtained by applying step 1 of TRW-S algorithm. Then there exists a subsequence $\{\boldsymbol{\theta}^{(i(m))}\}_m$ such that*

(a) It has reparameterization $\{\tilde{\boldsymbol{\theta}}^{i(m)}\}_m$ (i.e. $\tilde{\boldsymbol{\theta}}^{i(m)} \in \mathcal{A}$ and $(\theta^{i(m)})^T \equiv (\tilde{\theta}^{i(m)})^T$ for any m, T) that converges to some vector $\boldsymbol{\theta}^* \in \mathcal{A}$: $\{\tilde{\boldsymbol{\theta}}^{i(m)}\} \overset{m \to \infty}{\longrightarrow} \boldsymbol{\theta}^*$.

(b) Sequence $\{\Phi_\rho(\boldsymbol{\theta}^{(i)})\}_i$ converges to $\Phi_\rho(\boldsymbol{\theta}^*)$.

(c) Vector $\boldsymbol{\theta}^*$ satisfies WTA condition.

3.3 TRW-S algorithm for a graph with monotonic chains

In this section we focus on step 1(a) - reparameterizing vector θ^T. For simplicity, consider the case when $\omega = s$ is a node. In general, a complete forward pass of the ordinary max-product BP is needed for trees $T \in \mathcal{T}_s$ - sending messages from leaves to node s which we treat as a root[5]. However, this would make the algorithm very inefficient if trees are large. Fortunately, a complete forward pass is not always necessary.

The key idea is that the averaging operation does not invalidate messages in trees $T \in \mathcal{T}_s$ oriented *towards* node s[6]. Therefore, we can "reuse" messages passed in previous steps, i.e. not pass them again. This observation allows us to reduce the number of messages drastically if trees and the order of averaging operations are chosen in a particular way. Specifically, we require trees to be chains which are *monotonic* with respect to some ordering on the graph:

Definition 3.7. *Graph $\mathcal{G}$ and chains $T \in \mathcal{T}$ are said to be* monotonic *if there exists an ordering of nodes $i(u), u \in \mathcal{V}$ such that each chain T satisfies the following property: if $u^T_1, \ldots, u^T_{n(T)}$ are the consecutive nodes in the chain, then the sequence $i(u^T_1), \ldots, i(u^T_{n(T)})$ is monotonic.*

[5] Note that the backward pass (sending messages from the root to leaves) is not needed. It would convert vectors θ^T to normal forms but would not change vectors θ^T_s.

[6] Recall that our algorithm is message-free. The phrase "message is valid for directed edge $(s \to t)$ in tree T" means that sending a message from node s to node t as discussed in section 2.1 would not modify vector θ^T.

0. Initialize $\boldsymbol{\theta}$ so that $\boldsymbol{\theta} \in \mathcal{A}$ and $\sum_T \rho^T \theta^T \equiv \bar{\theta}$.

1. For nodes $t \in \mathcal{V}$ do the following operations in the order of increasing $i(t)$:

   (a) For every edge $(s,t) \in \mathcal{E}$ with $i(s) < i(t)$ do the following:

   - If $\mathcal{T}_{st}$ contains more than one chain then perform the averaging operation for edge (s,t).
   - For chains in $\mathcal{T}_{st}$ pass a message from s to t.

   (b) Perform the averaging operation for node t.

2. Reverse the ordering: set $i(u) := |\mathcal{V}| + 1 - i(u)$.

3. Check whether a stopping criterion is satisfied; if yes, terminate, otherwise go to step 1.

Figure 2: **TRW-S algorithm for a graph with monotonic chains.**

As an example, we could choose $\mathcal{T}$ to be the set of edges; it is easy to see that they are monotonic for any ordering of nodes. However, it might be advantageous to choose longer trees since the information might propagate faster through the graph.

The algorithm for a graph with monotonic chains is shown in Fig. 2. Its properties are summarized by the following lemma whose proof is given in [4].

Lemma 3.8. *Starting with the second pass, the following properties hold during step 1 for node t:*

(a) For each edge $(u,v) \in \mathcal{E}$ with $i(u) < i(v) < i(t)$ messages $(u \to v)$ in chains $T \in \mathcal{T}_{uv}$ are valid. This property also holds for node $v = t$ in the beginning and in the end of step 1(b).

(b) For each edge $(u,v) \in \mathcal{E}$ with $i(u) \geq i(t)$ messages $(v \to u)$ in chains $T \in \mathcal{T}_{uv}$ are valid.

In addition, property (a) holds during the first pass of the algorithm.

Thus, the algorithm in Fig. 2 is a special case of the algorithm in Fig. 1, if we treat the first pass of former algorithm as part of initialization of the latter one.

Efficient implementation The algorithm in Fig. 2 requires $O(|\mathcal{T}_s| \cdot |\mathcal{X}_s|)$ storage for node s and $O(|\mathcal{T}_{st}| \cdot |\mathcal{X}_s| \cdot |\mathcal{X}_t|)$ storage for edge (s,t). However, we can reduce it to $O(|\mathcal{X}_s|)$ and $O(|\mathcal{X}_s| + |\mathcal{X}_t|)$, respectively, using two ideas[7]. First, it can be seen that the algorithm maintains the following equalities: $\theta_s^T = \theta_s^{T'}$ for $T, T' \in \mathcal{T}_s$ and $\theta_{st}^T = \theta_{st}^{T'}$ for $T, T' \in \mathcal{T}_{st}$ (assuming that they hold after initialization). Second,

[7] We assume that storage required for vectors $\bar{\theta}_{st}$ is negligible. This holds for many energy functions used in practice, e.g. for functions with Potts terms.

0. Set all messages to zero.

1. For nodes $t \in \mathcal{V}$ do the following operation in the order of increasing $i(t)$:

   - For every edge $(s,t) \in \mathcal{E}$ with $i(s) < i(t)$ update message M_{st} as follows:
     - Compute $\theta_s = \frac{1}{\rho_s}\bar{\theta}_s + \sum_{(u,s)\in\mathcal{E}} \frac{\rho_{us}}{\rho_s} M_{us}$
     - Set $M_{st;k} := \min_j \{(\theta_{s;j} - M_{ts;j}) + \frac{1}{\rho_{st}}\bar{\theta}_{st;jk}\}$

2. Reverse the ordering: set $i(u) := |\mathcal{V}| + 1 - i(u)$.

3. Check whether a stopping criterion is satisfied; if yes, terminate, otherwise go to step 1.

Figure 3: **Efficient implementation of the algorithm in Fig. 2 using messages.**

reparameterizations θ^T can be stored using messages $M_{st} = \{M_{st;k} \,|\, k \in \mathcal{X}_t\}$ for directed edges $(s \to t) \in \mathcal{E}$, which correspond to vector $\boldsymbol{\theta}$ as follows:

$$\begin{aligned} \theta_t^T &= \tfrac{1}{\rho_t}\bar{\theta}_t + \textstyle\sum_{(s,t)\in\mathcal{E}} \tfrac{\rho_{st}}{\rho_t} M_{st} \\ \theta_{st;jk}^T &= \tfrac{1}{\rho_{st}}\bar{\theta}_{st;jk} - M_{st;k} - M_{ts;j} \end{aligned}$$

The resulting algorithm is shown in Fig. 3. Note that for many important choices of terms $\bar{\theta}_{st}$ message update in step 1 can be done very efficiently using distance transforms [2].

The message update rule of our algorithm is very similar to that of TRW-E algorithm (they would be identical if trees were spanning). However, TRW-E performs the updates in parallel, while we do it sequentially. As a result, we get convergence properties proved earlier.

4 Experimental results

We have compared four algorithms: ordinary max-product algorithm (BP) and three tree-reweighted algorithms (TRW-E, TRW-T, TRW-S). For TRW-E algorithm we also experimented with the damping parameter $\gamma \in (0,1]$; as reported in [7], TRW-E converges if sufficiently damped. We tuned this parameter for the experiments below. We used rectangular graphs with 4-neighborhood systems. The trees were horizontal and vertical chains with uniform probability. We treated them as two forests with probabilities 0.5; then we have $\rho_s = 1$ for nodes s and $\rho_{st} = 0.5$ for edges (s,t). We processed nodes in the raster order.

For BP algorithm we implemented a sequential update scheme whose order is similar to that of TRW-S: we pass messages from the top left corner to the bottom right corner and back. We experimented with the parallel update scheme and found that it was much slower.

The running time of the algorithms (number of iterations) was measured in terms of the number of passed messages divided by the number of directed edges in the graph.

For TRW algorithms we measured two quantities as functions of time: the value of $\Phi_\rho(\boldsymbol{\theta})$ and the value of the energy $E(\mathbf{x} \mid \bar{\theta})$. Note that the former is a lower bound on the optimal value of the energy, and the latter is an upper bound. Solution $\mathbf{x}$ for vector $\boldsymbol{\theta}$ was computed as follows: for each node s we determine label x_s minimizing $\sum_T \rho^T \theta_s^T(x_s)$. For BP algorithm, we can determine only one of these quantities, namely the value of the energy.

TRW algorithms were deemed to converge if we could find a consistent collection $\mathbb{S} = \{\mathbb{S}^T\}$ such that $E(\mathbf{x}^T \mid \theta^T) - \Phi(\theta^T) < 10^{-8}$ for any tree T and configuration $\mathbf{x}^T \in \mathbb{S}^T$. Note that checking this condition is expensive. In practice it may be better to use a test based on the behaviour of function Φ_ρ.

4.1 Synthetically generated problems

We have tested two types of problems: Ising model with attractive potentials and with mixed potentials. Single-node potentials were generated as independent gaussians: $\bar{\theta}_{s;0}, \bar{\theta}_{s;1} \sim \mathcal{N}(0, (0.25)^2)$. Pairwise potentials were set as follows: $\bar{\theta}_{st;00} = \bar{\theta}_{st;11} = 0$, $\bar{\theta}_{st;01} = \bar{\theta}_{st;10} = \lambda_{st}$ where λ_{st} was generated as $|\mathcal{N}(0,1)|$ for attractive potentials and as $\mathcal{N}(0,1)$ for mixed potentials. The size of the problem was 20x20. We tested the algorithms on 100 sample problems.

Attractive potentials First we tested the behaviour of TRW-E algorithm with respect to the damping parameter γ. For $\gamma = 1$, only 40% of the trials converged; however, for smaller values of γ the algorithm always converged. The smallest number of iterations was achieved at $\gamma = 0.96$. We used this value for the subsequent experiment.

Next we tested convergence rate of the four algorithms. TRW-S and BP converged in 100% of the cases, and TRW-T never converged (we observed that it went into a loop). The average number of iterations until convergence was as follows: 73.1 for TRW-E, 21.8 for TRW-S and 7.1 for BP. Note that it is not very fair to compare convergence for TRW algorithms and for BP, since convergence criteria are different. Next test provides a better comparison.

In this test we measured average values of lower bound $\Phi_\rho(\boldsymbol{\theta})$ and energy $E(\mathbf{x} \mid \bar{\theta})$ as functions of time for the first 50 iterations. Results are shown in Fig. 4(a,b). In most cases TRW-E and TRW-S found an optimal solution, and BP found an optimal solution for a smaller number of cases. TRW-S was decreasing the energy faster than the other algorithms. Surprisingly, TRW-T algorithm failed completely. Moreover, it demonstrated similar behaviour for the tests below so it is not shown.

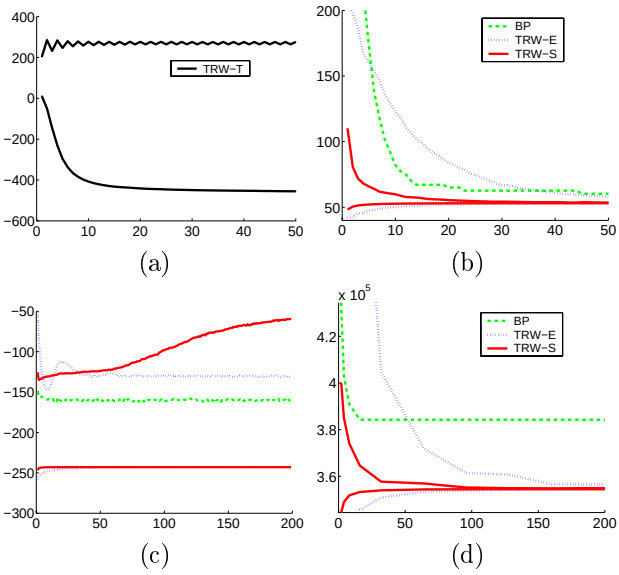


Figure 4: **Energy plots. Horizontal axis: number of iterations. Vertical axis, upper curves: average value of the energy $E(\mathbf{x} \mid \bar{\theta})$. Vertical axis, lower curves for TRW algorithms: average value of $\Phi_\rho(\boldsymbol{\theta})$. (a) Ising model, attractive potentials, TRW-T algorithm. (b) Ising model, attractive potentials, three other algorithms. (c) Ising model, mixed potentials. (d) Tsukuba dataset, Potts model.**

Mixed potentials The behaviour of TRW-E and TRW-S algorithms was substantially different from the previous case. We observed that the smallest value of the energy was achieved after a small number of iterations (5-10), and after that the energy increased. TRW-E with damping and TRW-S always converged, however the value of the energy at convergence was larger than in the middle. Moreover, the energy obtained by BP was smaller than the energy of TRW algorithms, despite the fact BP did not converge. This behaviour is shown in Fig. 4(c). For TRW-E algorithm we picked empirically $\gamma = 0.3$.

Poor performance of TRW algorithms for the problem with mixed potentials needs further investigation.

4.2 Stereo matching problem

We have tested the algorithms on the energy function arising in the stereo matching problem [1]. The input is two images taken from different viewpoints, and the goal is to find a disparity for every pixel in the left image. Similarly to [1], we used Potts interaction model.

For TRW-E algorithm we picked empirically $\gamma = 0.95$.

Fig. 4(d) shows energy plots for the Tsukuba dataset used in [1]. Precise numbers are as follows:

alg.	number of iterations				
	4	16	64	256	1024
expansion move [1]	719				
BP	48479	29815	29815	29815	29815
TRW-E	1073357	175433	17475	905	889
	-29418.0	-8966.5	-1058.7	-26.8	-0.47
TRW-S	30663	10101	2465	463	643
	-5496.0	-1189.0	-92.7	-0.057	0

Top rows show the value of the energy, bottom rows for TRW-E and TRW-S - the value of lower bound $\Phi_\rho(\boldsymbol{\theta})$. Note that we subtracted 354453 from all values - it appears to be the maximum of problem 4. Using a different initialization for TRW-S algorithm, we were also able to obtain a configuration with energy 354453+103.

TRW-S algorithm is a clear winner - it decreases the energy more rapidly than the other algorithms. The second best is TRW-E algorithm with damping, although in the beginning the energy decreases more slowly than the energy for BP. Our algorithm obtains lower energy than BP, and slightly lower energy than the expansion move algorithm [1].

5 Discussion and conclusions

We have developed a new algorithm which can be viewed as a method for direct maximization of objective function Φ_ρ subject to the constraint of problem 4. We gave a precise characterization of local maxima of this function with respect to TRW-S algorithm. We showed that the algorithm is guaranteed to have a subsequence converging to such a maximum.

As all tree-reweighted algorithms, our method is not guaranteed to find a *global* maximum. Nevertheless, experimental results suggest that this is not an issue for certain synthetic and real problems. For the stereo matching problem we were able to obtain slightly lower energy than the expansion move algorithm [1] which is considered to be the most accurate energy minimization technique for such problems. TRW-S algorithm outperforms both TRW algorithms in [7] and the ordinary max-product BP.

Our algorithm can be implemented efficiently only for a particular choice of trees. Fortunately, nothing prevents us from using this choice - the optimal value of the lower bound does not depend on it.

In our experiments we noticed that TRW-S algorithm would always converge to a fixed point of TRW, although such convergence would usually take much longer than achieving a weak tree agreement. However, we have not been able to prove this general convergence. On the other hand, in practice it does not make much sense to run the algorithm after WTA has been achieved since function Φ_ρ and, more importantly, the output configuration will not change.

In the future we plan to extend techniques proposed in this paper to sum-product TRW algorithm [9].

Acknowledgements

I would like to thank Thomas Minka for helpful discussions.

References

[1] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11), November 2001.

[2] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient belief propagation for early vision. *CVPR*, 2004.

[3] T. Heskes, K. Albers, and B. Kappen. Approximate inference and constrained optimization. *UAI*, 2003.

[4] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. Technical Report MSR-TR-2004-90, Misrosoft Research, September 2004.

[5] Y.W. Teh and M. Welling. The unified propagation and scaling algorithm. *NIPS*, 2002.

[6] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. A new class of upper bounds on the log partition function. *UAI*, 2002.

[7] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. MAP estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. Technical Report UCB/CSD-03-1269, UC Berkeley CS Division, August 2003.

[8] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, 45(9):1120–1146, May 2003.

[9] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. Tree-reweighted belief propagation and approximate ML estimation by pseudo-moment matching. *AISTATS*, 2003.

[10] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. *Statistics and Computing*, 14(2):143–166, April 2004.

[11] W. Wiegerinck and T. Heskes. Fractional belief propagation. *NIPS*, 2000.

[12] J.S. Yedidia, W.T. Freeman, and Y.Weiss. Generalized belief propagation. *NIPS*, 2000.

[13] A.L. Yuille. CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation. *Neural Computation*, 14:1691–1722, 2002.

Instrumental Variable Tests for Directed Acyclic Graph Models

Manabu Kuroki
Department of Systems Innovation
Graduate School of Engineering Science
Osaka University
Machikaneyama-cho, Toyonaka, Osaka
mkuroki@sigmath.es.osaka-u.ac.jp

Zhihong Cai
Department of Biostatistics
Graduate School of Public Health
Kyoto University
Yoshida, Konoe-cho, Sakyo-ku, Kyoto
cai@pbh.med.kyoto-u.ac.jp

Abstract

Consider a case where cause-effect relationships among variables can be described as a directed acyclic graph model. The instrumental variable (IV) method is well known as a powerful tool for inferring the total effect of a treatment variable X on a response variable Y, even if there exist unobserved confounders between X and Y. This paper proposes a criterion to search for covariates which satisfy the IV conditions in linear structural equation models. In addition, it is shown that our criterion is applicable to the case where all variables in a directed acyclic graph are discrete. The results of this paper provide a statistical justification for testing whether the statistical data is generated by a model involving IVs.

1 INTRODUCTION

Estimating total effects is an important problem in many scientific domains. The back door criterion (e.g. Pearl, 2000) and the instrumental variable (IV) method (e.g. Bowden and Turkington, 1984) are two main powerful tools to estimate total effects. However, in the case where unobserved confounders exist, it may be difficult to apply the back door criterion to estimating total effects. Under such circumstances, it is beneficial to utilize the IV method. Here, a total effect τ_{yx} of X on Y is a measure for evaluating causal effects, and can be interpreted as the change of the mean of a response variable Y when a treatment variable X is changed by one unit through an external intervention.

The IV method is well known as a powerful tool to estimate total effects from observational data, in case where unobserved confounders are at presence. Therefore, the IV method has been studied by many researchers in practical sciences (e.g. Angrist et al, 1996; Bowden and Turkington, 1984; Greenland, 2000). It is also one of current hot topics in artificial intelligence and statistics (e.g. Bollen and Bauer, 2004; Brito and Pearl, 2002; Chu et al., 2001; Kuroki and Cai, 2004; Lauritzen, 2004; Pearl, 2004; Scheines et al., 2001). In the IV method, when we consider a linear regression model $Y = \tau_{yx}X + U$ in the case where an unobserved confounder U is correlated with X, the total effect τ_{yx} can be estimated through an observed variable Z (called an IV) which is correlated with X but not with U, and is given by σ_{yz}/σ_{xz}. Here, σ_{yz} and σ_{xz} are a covariance between Y and Z and a covariance between X and Z, respectively.

As many researchers have pointed out, a serious problem of the IV method is that it is difficult to test whether a covariate satisfies the IV conditions from observational data, since U is an unobserved variable. In order to solve this problem, Pearl (1995) provided the instrumental inequality as a necessary condition to test whether a variable Z is an IV relative to (X, Y), in the case where X is a discrete variable. However, there is no testing method for IVs when X is continuous. Considering that there are many applications regarding the IV methods in linear structural equation models, it is necessary to develop a testing criterion in this framework.

In this paper, we first introduce the tetrad difference which has been used to test whether statistical dependencies among observed variables are due to a single common factor in factor analysis. It is shown that the tetrad difference can be applied to linear case as well as discrete case. Although searching for one latent common factor is quite different from searching for an IV, we will show that similar consideration is helpful to provide a criterion to search for covariates which satisfy the IV conditions. Pearl (1995) provided an inequality to search for one instrumental variable, while we propose an equation to search for two or more IVs. Hence, our method may provide a tighter condi-

tion than Pearl (1995). Since it is the usual case in practical science that there exist many observed covariates as well as unobserved confounders, if we can find two or more covariates satisfying the IV conditions, it is possible to apply IV method to estimating total effects, whether such covariates are continuous or discrete.

This paper is arranged as follows. In the next section, basic graph terminologies are given. Section 3 describes previous studies about the tetrad difference, and then we show that tetrad difference is applicable to discrete case. On the basis of similar consideration, section 4 provides testing criteria for conditional IVs (Brito and Pearl, 2002) in both continuous case and discrete case. Section 5 illustrates our results with an example. Some further topics are presented in the final section.

2 PRELIMINARIES

2.1 GRAPHS

A directed graph is a pair $G = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V}$ is a finite set of vertices and the set $\boldsymbol{E}$ of arrows is a subset $\boldsymbol{V}\times\boldsymbol{V}$ of ordered pairs of distinct vertices. An arrow pointing from a vertex a to a vertex b indicates $(a,b)\in\boldsymbol{E}$ and $(b,a) \notin \boldsymbol{E}$. The arrow is said to emerge from a or point to b. If there is an arrow pointing from a to b, a is said to be a parent of b, and b a child of a. The set of parents of b is denoted as $pa(b)$, and the set of children of a as $ch(a)$.

A path between a and b is a sequence $a = a_0$, a_1, $\cdots$, $b = a_n$ of distinct vertices such as $(a_{i-1}, a_i)\in\boldsymbol{E}$ or $(a_i, a_{i-1})\in\boldsymbol{E}$ for all $i = 1, 2, \cdots, n$. A directed path from a to b is a sequence $a = a_0$, a_1, $\cdots$, $b = a_n$ of distinct vertices such as $(a_{i-1}, a_i)\in\boldsymbol{E}$ and $(a_i, a_{i-1})\notin\boldsymbol{E}$ for all $i = 1, \cdots, n$. If there exists a directed path from a to b, a is said to be an ancestor of b and b a descendant of a. When the set of descendants of a is denoted as $de(a)$, the vertices in $\boldsymbol{V}\backslash(de(a)\cup\{a\})$ are said to be the nondescendants of a. A directed cycle is a sequence a_0, a_1, $\cdots$, a_{n-1} of distinct vertices such as (1) $(a_{i-1}, a_i)\in\boldsymbol{E}$ and $(a_i, a_{i-1}) \notin\boldsymbol{E}$ for all $i = 1, \cdots, n-1$, and (2) $(a_{n-1}, a_0)\in\boldsymbol{E}$.

If a directed graph has no directed cycles, then the graph is said to be a directed acyclic graph.

2.2 BAYESIAN NETWORKS

Let $\mathrm{p}_{v_1,\cdots,v_n}$ be a strictly positive joint distribution of $\boldsymbol{V} = \{V_1, \cdots, V_n\}$ and $\mathrm{p}_{v_i|v_j}$ the conditional distribution of V_i given V_j. Similar notations are used for other distributions. Then, a graphical representation of the conditional independencies among $V_1, \cdots, V_n$ in form of a directed acyclic graph is given in the following way:

DEFINITION 1 (BAYESIAN NETWORK)
Suppose that a set of variables $\boldsymbol{V} = \{V_1, \ldots, V_n\}$ and a directed acyclic graph $G = (\boldsymbol{V}, \boldsymbol{E})$ are given. When the joint distribution of $\boldsymbol{V}$ is factorized recursively according to the graph G as the following equation, the graph is called a Bayesian network.

$$\mathrm{p}_{v_1,\cdots,v_n} = \prod_{i=1}^{n} \mathrm{p}_{v_i|pa(v_i)}, \tag{1}$$

When $pa(V_i)$ is an empty set, $\mathrm{p}_{v_i|pa(v_i)}$ is the marginal distribution of v_i. □

If a joint distribution is factorized recursively according to the graph G, the conditional independencies implied by the factorization (1) can be obtained from the graph G according to the following criterion (Pearl, 1988):

DEFINITION 2 (D-SEPARATION)
Let $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ be three disjoint subsets of vertices in a Bayesian network G. Then $\boldsymbol{Z}$ is said to d-separate $\boldsymbol{X}$ from $\boldsymbol{Y}$, if along every path between a vertex in $\boldsymbol{X}$ and a vertex in $\boldsymbol{Y}$ there exists a vertex w which satisfies one of the following three conditions:

1. w is in $\boldsymbol{Z}$, and one arrow on the path points to w and the other arrow on the path emerges from w.
2. w is in $\boldsymbol{Z}$, and two arrows on the path emerge from w.
3. Neither w nor any descendant of w is in $\boldsymbol{Z}$, but two arrows on the path point to w.

If a path satisfies one of the conditions above, the path is said to be blocked by $\boldsymbol{Z}$. □

It can be shown that if $\boldsymbol{Z}$ d-separates $\boldsymbol{X}$ from $\boldsymbol{Y}$ in a Bayesian network G then $\boldsymbol{X}$ is conditionally independent of $\boldsymbol{Y}$ given $\boldsymbol{Z}$ in the corresponding recursive factorization (1) (Geiger et al., 1990).

2.3 LINEAR STRUCTURAL EQUATION MODEL

In the case where variables in $\boldsymbol{V}$ are normally distributed, equation (1) provides a linear structural equation model

$$V_i = \sum_{V_j\in pa(V_i)} \alpha_{v_i v_j} V_j + \epsilon_{v_i} \qquad i = 1, \ldots, n, \tag{2}$$

where $\alpha_{v_i v_j} (\neq 0)$ is called a path coefficient. In the linear structural equation models discussed in this paper, it is assumed that each variable in $\boldsymbol{V}$ has mean 0, and

that random disturbances $\epsilon_{v_1}, \cdots, \epsilon_{v_n}$ are independent and normally distributed. For further details of linear structural equation models, see Bollen (1989).

Let $\sigma_{xy \cdot z}$ and $\beta_{yx \cdot z}$ be the conditional covariance of X and Y given $\boldsymbol{Z}$ and the regression coefficient of x in the regression model of Y on x and $\boldsymbol{z}$, respectively. $\rho_{xy \cdot z}$ is a partial correlation coefficient of X and Y given $\boldsymbol{Z}$, and $\sigma_{yy \cdot z}$ is a conditional variance of Y given $\boldsymbol{Z}$. In the case where $\boldsymbol{Z}$ is an empty set, σ_{xy} and β_{yx} are the covariance of X and Y and the regression coefficient of x in the regression model of Y on x, respectively. In addition, ρ_{xy} and σ_{yy} are the correlation coefficient of X and Y and the variance of Y, respectively. Similar notations are used for other parameters.

When a linear structural equation model is given according to the graph G, if $\boldsymbol{Z}$ d-separates X from Y, then $\sigma_{xy \cdot z} = \beta_{yx \cdot z} = 0$ (e.g. Spirtes et al., 1993).

3 TETRAD DIFFERENCE

3.1 LINEAR CASE

We consider the directed acyclic graph shown in Fig.1 and the corresponding linear structural equation model.

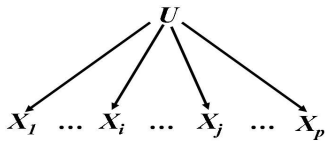


Fig.1: Directed acyclic graph (1)

As shown in Fig.1, U is a parent of each element of $\boldsymbol{X} = (X_1, \cdots, X_p)$ and d-separates any two elements of $\boldsymbol{X}$. The covariance structure corresponding to Fig.1 can be described as

$$\Sigma_{xx} = \Sigma_{xx \cdot u} + \frac{1}{\sigma_{uu}} \Sigma_{xu} \Sigma'_{xu}, \tag{3}$$

where Σ_{xx} and $\Sigma_{xx \cdot u}$ is a covariance matrix of $\boldsymbol{X}$ and a conditional covariance matrix of $\boldsymbol{X}$ given U, respectively. In addition, Σ_{xu} is a covariance matrix between $\boldsymbol{X}$ and U. Similar notations are used for other matrices.

Then, it can be understood that

1) $\Sigma_{xx \cdot u}$ is a positive diagonal matrix which satisfies $\mathrm{rank}(\Sigma_{xx} - \Sigma_{xx \cdot u}) = 1$ and $\Sigma_{xx} - \Sigma_{xx \cdot u}$ is a positive semidefinite matrix, and

2) Since $\sigma_{x_i x_j} \sigma_{x_k x_l} = (\sigma_{x_i u} \sigma_{x_j u} / \sigma_{uu})(\sigma_{x_k u} \sigma_{x_l u} / \sigma_{uu}) = \sigma_{x_i x_k} \sigma_{x_j x_l}$,

$$\sigma_{x_i x_j} \sigma_{x_k x_l} - \sigma_{x_i x_k} \sigma_{x_j x_l} = 0 \tag{4}$$

$$\sigma_{x_i x_i} - \frac{\sigma_{x_i x_j} \sigma_{x_i x_k}}{\sigma_{x_j x_k}} = \sigma_{x_i x_i} - \frac{\sigma^2_{x_i u}}{\sigma_{uu}} > 0$$

can be obtained.

For any distinct variables $X_i, X_j, X_k, X_l \in \boldsymbol{X} (\subset \boldsymbol{V})$, Spearman (1904, 1928) defined the tetrad difference as the left hand side of equation (4). When the tetrad difference is equal to zero, it is called a vanishing tetrad difference. It is stated that the vanishing tetrad difference implies that the observed statistical dependencies can be well explained by a single factor model $X_i = \lambda_i U + \epsilon_{x_i}$ $(i = 1, \cdots, p)$, where λ_i is a constant value and $\epsilon_{x_1}, \cdots, \epsilon_{x_p}$ are independent.

The tetrad difference has been studied by many researchers for decades. Bekker and de Leeuw (1987) summarized the previous results into a single comprehensive theorem. To present this theorem, the following preliminaries are required. When there exists such a positive semidefinite diagonal matrix Ω that $\Sigma_{xx} - \Omega$ is a positive semidefinite matrix and $\mathrm{rank}(\Sigma_{xx} - \Omega) = 1$, Σ_{xx} is said to be a Spearman matrix, which provides statistical evidence that the observed statistical dependencies can be well explained by a single factor model (Bekker and de Leeuw, 1987). In addition, any element of Σ_{xx} is assumed to be nonzero. With this preparation, Bekker and de Leeuw (1987) provided the following theorem:

THEOREM 1 (Bekker and de Leeuw, 1987)
A covariance matrix $\Sigma_{xx} = (\sigma_{x_i x_j})$ of a set $\boldsymbol{X}$ including four or more variables is a Spearman matrix if, and only if, after sign changes of rows and corresponding columns, all its elements are positive and such that

$$\sigma_{x_i x_j} \sigma_{x_k x_l} - \sigma_{x_i x_k} \sigma_{x_j x_l} = 0 \tag{5}$$

and

$$\frac{\sigma_{x_i x_j} \sigma_{x_i x_k}}{\sigma_{x_j x_k}} \leq \sigma_{x_i x_i} \tag{6}$$

for any distinct variables $X_i, X_j, X_k, X_l \in \boldsymbol{X}$. □

If Theorem 1 holds true, then we can justify the assumption that the observed statistical dependencies can be generated by a single common factor. Regarding the further discussion of models implied by Theorem 1, see Bollen and Ting (2000), Spirtes et al. (1993) and Xu and Pearl (1987).

3.2 DISCRETE CASE

In this section, we show that the similar result of Theorem 1 holds true in the case where all variables in a directed acyclic graph are dichotomous variables.

For dichotomous variables $X, Y \in \boldsymbol{V} (a, b \in \{0, 1\})$, let $\mathrm{p}_{X=a, Y=b}$ be a joint probability of $X = a$ and $Y = b$, and $\mathrm{p}_{X=a}$ a marginal probability of $X = a$. In addition, $\mathrm{p}_{X=a|Y=b}$ is a conditional probability of $X = a$ given $Y = b$ $(a, b \in \{0, 1\})$. Similar notations are used for other parameters.

Letting $\sigma_{x_i x_j} = \mathrm{p}_{x_i=1,x_j=1} - \mathrm{p}_{x_i=1}\mathrm{p}_{x_j=1}$ and $\sigma_{x_i x_i} = \mathrm{p}_{x_i=1}(1-\mathrm{p}_{x_i=1})$ (These are correspondent to a covariance between X_i and X_j and a variance of X_i, respectively), based on the Bayesian network shown in Fig.1, since

$$\begin{aligned}\sigma_{x_i x_j} &= (\mathrm{p}_{x_i=1|u=0} - \mathrm{p}_{x_i=1|u=1}) \\ &\quad \times(\mathrm{p}_{x_j=1|u=0} - \mathrm{p}_{x_j=1|u=1})\mathrm{p}_{u=0}\mathrm{p}_{u=1}\end{aligned}$$

for any X_i and X_j, we can obtain

$$\begin{aligned}\Sigma_{xx} &= \mathrm{Diag}[\omega_{11},\cdots,\omega_{pp}] \\ &\quad +\mathrm{p}_{u=0}\mathrm{p}_{u=1}\begin{pmatrix} \mathrm{p}_{x_1=1|u=0} - \mathrm{p}_{x_1=1|u=1} \\ \vdots \\ \mathrm{p}_{x_p=1|u=0} - \mathrm{p}_{x_p=1|u=1}\end{pmatrix} \\ &\quad \times(\mathrm{p}_{x_1=1|u=0} - \mathrm{p}_{x_1=1|u=1},\cdots,\mathrm{p}_{x_p=1|u=0} - \mathrm{p}_{x_p=1|u=1}).\end{aligned}$$

Here,

$$\omega_{ii} = \sigma_{x_i x_i} - (\mathrm{p}_{x_i=1|u=0} - \mathrm{p}_{x_i=1|u=1})^2\mathrm{p}_{u=0}\mathrm{p}_{u=1}$$

for $i = 1, 2, \cdots, p$. Then, since

$$\begin{aligned}\omega_{ii} &= \mathrm{p}_{u=0}(\mathrm{p}_{x_i=1|u=0} - \mathrm{p}^2_{x_i=1|u=0}) \\ &\quad +\mathrm{p}_{u=1}(\mathrm{p}_{x_i=1|u=1} - \mathrm{p}^2_{x_i=1|u=1}) \geq 0\end{aligned}$$

for any X_i, we can understand that Σ_{xx} can be also expressed as the form of $\Omega + \boldsymbol{\lambda}\boldsymbol{\lambda}'$, where Ω is a positive definite diagonal matrix and $\boldsymbol{\lambda}$ is a vector.

Thus, based on the consideration above, it is shown that the same result as Theorem 1 holds true in the case where all variables in a directed acyclic graph are dichotomous. Regarding the further discussion, see Pearl and Tarci (1986).

4 INSTRUMENTAL VARIABLE (IV) METHOD

4.1 IDENTIFICATION

In this section, we introduce the conditional instrumental variable (IV) method (Brito and Pearl, 2002) as the identifiability criterion for total effects. Here, a total effect τ_{yx} of X on Y is defined as the total sum of the products of the path coefficients on the sequence of arrows along all directed paths from X to Y. When a total effect can be determined uniquely from the correlation parameters of observed variables, it is said to be identifiable, that is, it can be estimated consistently. Let $G_{\underline{X}}$ be the graph obtained by deleting from a graph G all arrows emerging from vertices in $\boldsymbol{X}$.

DEFINITION 3 (CONDITIONAL IV)
If a set $\boldsymbol{W} \cup \{Z\}$ of variables satisfies the following conditions relative to an ordered pair (X, Y) of variables in a directed acyclic graph G, then Z is said to be a conditional instrumental variable (IV) given $\boldsymbol{W}$ relative to (X, Y).

1. A set $\boldsymbol{W}$ of variables is a subset of nondescendants of X and Y in G, and
2. $\boldsymbol{W}$ d-separates Z from Y but not from X in $G_{\underline{X}}$.

□

In linear structural equation models, if an observed variable Z is a conditional IV given a set $\boldsymbol{W}$ of observed variables relative to (X, Y), then the total effect τ_{yx} of X on Y is identifiable, and is given by $\sigma_{yz\cdot w}/\sigma_{xz\cdot w}$ (Brito and Pearl, 2002). When $\boldsymbol{W}$ is an empty set in Definition 3, Z is called an instrumental variable (IV) (Bowden and Turkington, 1984).

When the sample conditional covariances $\hat{\sigma}_{yz\cdot w}$ and $\hat{\sigma}_{xz\cdot w}$ of the conditional covariances $\sigma_{yz\cdot w}$ and $\sigma_{xz\cdot w}$ are used to estimate the total effect τ_{yx}, by the delta method (Anderson, 1986), the asymptotic variance of $\hat{\sigma}_{yz\cdot w}/\hat{\sigma}_{xz\cdot w}$ is given by

$$a.var\left(\frac{\hat{\sigma}_{yz\cdot w}}{\hat{\sigma}_{xz\cdot w}}\right) = \frac{\sigma_{yy\cdot w}/\sigma_{xx\cdot w} - 2\beta_{yx\cdot w}\tau_{yx} + \tau^2_{yx}}{n\rho^2_{xz\cdot w}} \quad (7)$$

(e.g. Kuroki and Cai, 2004), where n is the sample size, and $\rho_{xz\cdot w}$ is a partial correlation coefficient of X and Z given $\boldsymbol{W}$. Clearly, for a given $\boldsymbol{W}$, the smaller the value of the $\rho^2_{xz\cdot w}$ is, the larger the asymptotic variance of $\hat{\sigma}_{yz\cdot w}/\hat{\sigma}_{xz\cdot w}$ is. This fact provides an IV selection criterion for estimating total effects (e.g. Kuroki and Cai, 2004).

4.2 LINEAR CASE

In this section, we propose a criterion for testing whether statistical data is generated by a model involving IVs. For this purpose, consider the directed acyclic graph shown in Fig.2.

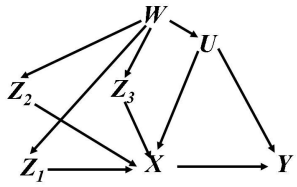


Fig.2: Directed acyclic graph (2)

Fig.2 shows that Z_1, Z_2 and Z_3 are conditional IVs given $\boldsymbol{W}$ relative to (X, Y). In addition, $Z_i \perp\!\!\!\perp Z_j | \boldsymbol{W}$ holds true $(i, j = 1, 2, 3 (i \neq j))$. Then, the following conditional covariances can be obtained:

$$\begin{aligned}\sigma_{z_i z_j\cdot xw} &= -\frac{\sigma_{z_i x\cdot w}\sigma_{z_j x\cdot w}}{\sigma_{xx\cdot w}} \qquad (i, j = 1, 2, 3 (i \neq j)) \\ \sigma_{z_i y\cdot xw} &= \sigma_{z_i y\cdot w} - \frac{\sigma_{z_i x\cdot w}\sigma_{yx\cdot w}}{\sigma_{xx\cdot w}} \qquad (i = 1, 2, 3).\end{aligned}$$

Here, by $\tau_{yx} = \sigma_{yz_i \cdot w}/\sigma_{xz_i \cdot w} (i = 1, 2, 3)$, we can obtained

$$\sigma_{z_i y \cdot xw} = \sigma_{xz_i \cdot w}(\tau_{yx} - \beta_{yx \cdot w}).$$

Hence, letting $\boldsymbol{Z} = \{Z_1, Z_2, Z_3\}$, the conditional covariance matrix of $\boldsymbol{S} = \boldsymbol{Z} \cup \{Y\}$ given X and $\boldsymbol{W}$ can be provided as

$$\begin{aligned}
&\Sigma_{ss \cdot xw} \\
&= \begin{pmatrix} \Sigma_{zz \cdot w} & \mathbf{0} \\ \mathbf{0}' & \sigma_{yy \cdot xw} + \sigma_{xx \cdot w}(\tau_{yx} - \beta_{yx \cdot w})^2 \end{pmatrix} \\
&\quad - \begin{pmatrix} \frac{\Sigma_{zx \cdot w}}{\sqrt{\sigma_{xx \cdot w}}} \\ -\sqrt{\sigma_{xx \cdot w}}(\tau_{yx} - \beta_{yx \cdot w}) \end{pmatrix} \\
&\quad \times \left(\frac{\Sigma'_{zx \cdot w}}{\sqrt{\sigma_{xx \cdot w}}}, -\sqrt{\sigma_{xx \cdot w}}(\tau_{yx} - \beta_{yx \cdot w})\right) \\
&= \Omega - \boldsymbol{\lambda}\boldsymbol{\lambda}', \qquad (8)
\end{aligned}$$

where $\Sigma'_{xz \cdot w}$ is a conditional covariance vector of X and $\boldsymbol{Z}$ given $\boldsymbol{W}$. Similar notations are used for other vectors. That is, a necessary condition that Z_1, Z_2 and Z_3 are conditional IVs given $\boldsymbol{W}$ relative to (X, Y) is that $\Sigma_{ss \cdot xw}$ can be reformulated as equation (8) through a positive diagonal matrix Ω and a vector $\boldsymbol{\lambda}$. This necessary condition, which is testable by observational data, holds true when there exist two or more covariates which satisfy the IV conditions. In addition, from equation (8), it can be understood that

1) $\sigma_{z_1 z_2 \cdot xw}\sigma_{z_3 y \cdot xw} = \sigma_{z_1 z_3 \cdot xw}\sigma_{z_2 y \cdot xw} = \sigma_{z_2 z_3 \cdot xw}\sigma_{z_1 y \cdot xw}$ hold true in Fig.2, which is similar to the vanishing tetrad differences.

2) when the (i, j) component of $\Sigma_{ss \cdot xw}$ in equation (8) is positive, letting Q be the matrix obtained from the unit matrix by multiplying the i th or j th diagonal component with -1, we can obtain

$$Q\Sigma_{ss \cdot xw}Q' = Q\Omega Q' - (Q\boldsymbol{\lambda})(Q\boldsymbol{\lambda})'.$$

That is, by sign changes of rows and corresponding columns, all the off-diagonal elements of $\Sigma_{ss \cdot xw}$ in equation (8) can become negative, since multiplications of the column vector $\boldsymbol{\lambda}$ with Q finally turn all the elements of $Q\boldsymbol{\lambda}$ into positive.

3) $\Sigma_{ss \cdot xw} - \Omega$ is a negative semidefinite matrix and $\mathrm{rank}(\Sigma_{ss \cdot xw} - \Omega) = 1$.

4) the (i, i) component of Ω is equal to the conditional variance of Z_i given $\boldsymbol{W}$ $(i = 1, 2, 3)$ but the $(4, 4)$ component of Ω is not equal to the conditional variance of Y given $\boldsymbol{W}$.

5) When $\tau_{yx} = \beta_{yx \cdot w}$ holds true, the last component corresponding to Y in $\boldsymbol{\lambda}$ is equal to zero. This occurs in the case where $\boldsymbol{W}$ satisfies "the back door criterion" relative to (X, Y) (for "the back door criterion", refer to Pearl (2000)).

On the basis of the considerations above, when any element of $\Sigma_{ss \cdot xw}$ is assumed to be nonzero and $\beta_{yx \cdot w}$ is not consistent with τ_{yx}, the following theorem can be obtained:

THEOREM 2

For a covariance matrix $\Sigma_{ss \cdot xw}$ of a set $\boldsymbol{S} = \{Z_1, \cdots, Z_p\}$ (here, letting $Z_p = Y$) including four or more variables, after sign changes of rows and corresponding columns of $\Sigma_{ss \cdot xw}$, all its off-diagonal elements are negative and such that

$$\sigma_{z_i z_j \cdot xw}\sigma_{z_k z_l \cdot xw} = \sigma_{z_i z_k \cdot xw}\sigma_{z_j z_l \cdot xw} \qquad (9)$$

and

$$\frac{\sigma_{z_i z_k \cdot xw}\sigma_{z_i z_j \cdot xw}}{\sigma_{z_j z_k \cdot xw}} \leq \sigma_{z_i z_i \cdot xw} \qquad (10)$$

for any distinct variables $Z_i, Z_j, Z_k, Z_l \in \boldsymbol{S}$ if, and only if, there exist a positive definite diagonal matrix Ω and a vector $\boldsymbol{\lambda}$ satisfying that

$$\Sigma_{ss \cdot xw} = \Omega - \boldsymbol{\lambda}\boldsymbol{\lambda}'. \qquad (11)$$

□

PROOF OF THEOREM 2

First, suppose that there exist a positive definite diagonal matrix Ω and a vector $\boldsymbol{\lambda}$, which satisfies equation (11). Then, from the consideration 2) stated above, it is trivial that all its off-diagonal elements can be negative after sign changes of rows and corresponding columns of $\Sigma_{ss \cdot xw}$. In addition, since $\sigma_{z_i z_j \cdot xw} = -\lambda_i\lambda_j$ holds true for any Z_i and Z_j, we can obtain

$$\begin{aligned}
&\sigma_{z_i z_j \cdot xw}\sigma_{z_k z_l \cdot xw} = (-\lambda_i\lambda_j)(-\lambda_k\lambda_l) \\
&= \sigma_{z_i z_k \cdot xw}\sigma_{z_l z_j \cdot xw} = \sigma_{z_i z_l \cdot xw}\sigma_{z_k z_j \cdot xw}
\end{aligned}$$

and

$$\begin{aligned}
&\sigma_{z_i z_i \cdot xw} - \frac{\sigma_{z_i z_j \cdot xw}\sigma_{z_i z_k \cdot xw}}{\sigma_{z_j z_k \cdot xw}} \\
&= \sigma_{z_i z_i \cdot xw} - \frac{(-\lambda_i\lambda_j)(-\lambda_i\lambda_k)}{(-\lambda_j\lambda_k)} \\
&= \sigma_{z_i z_i \cdot xw} + \lambda_i^2 > 0.
\end{aligned}$$

Thus, equations (9) and (10) can be obtained from $\Sigma_{ss \cdot xw}$.

Next, as we can assume that all off-diagonal elements of $\Sigma_{ss \cdot xw}$ are negative, by sign changes of rows and corresponding columns of $\Sigma_{ss \cdot xw}$, it will be shown that a p dimensional vector $\boldsymbol{\lambda}' = (\lambda_1, \cdots, \lambda_p)$ exists such that $\Sigma_{ss \cdot xw} + \boldsymbol{\lambda}\boldsymbol{\lambda}'$ is a positive diagonal matrix. Each element of $\sigma_{z_i z_j \cdot xw}$ $(Z_i \neq Z_j)$ can be described as

$$\begin{aligned}
\sigma_{z_i z_j \cdot xw} &= \frac{\sigma_{z_i z_k \cdot xw}\sigma_{z_j z_l \cdot xw}}{\sigma_{z_k z_l \cdot xw}} \\
&= -\left(\left|\frac{\sigma_{z_i z_k \cdot xw}\sigma_{z_i z_j \cdot xw}}{\sigma_{z_j z_k \cdot xw}}\right|\right)^{1/2} \left(\left|\frac{\sigma_{z_j z_k \cdot xw}\sigma_{z_i z_j \cdot xw}}{\sigma_{z_i z_k \cdot xw}}\right|\right)^{1/2} \\
&= -\lambda_i\lambda_j.
\end{aligned}$$

Here, from equation (9), it is noted that $\lambda_i = (|\sigma_{z_i z_k \cdot xw}\sigma_{z_i z_j \cdot xw} / \sigma_{z_j z_k \cdot xw}|)^{1/2}$ takes the same value regardless of the choice of Z_j and Z_k, since

$$\frac{\sigma_{z_i z_k \cdot xw}\sigma_{z_i z_j \cdot xw}}{\sigma_{z_j z_k \cdot xw}} = \frac{\sigma_{z_i z_l \cdot xw}\sigma_{z_i z_j \cdot xw}}{\sigma_{z_j z_l \cdot xw}} = \frac{\sigma_{z_i z_l \cdot xw}\sigma_{z_i z_m \cdot xw}}{\sigma_{z_m z_l \cdot xw}},$$

(Z_i, Z_j, Z_k, Z_l and Z_m are distinct). On the other hand, noting that equation (10) is automatically satisfied since all off-diagonal elements of $\Sigma_{ss \cdot xw}$ are negative,

$$\begin{aligned}\sigma_{z_i z_i \cdot xw} &= (\sigma_{z_i z_i \cdot xw} + \left|\frac{\sigma_{z_i z_k \cdot xw}\sigma_{z_i z_j \cdot xw}}{\sigma_{z_j z_k \cdot xw}}\right|) \\ &\quad - \left|\frac{\sigma_{z_i z_k \cdot xw}\sigma_{z_i z_j \cdot xw}}{\sigma_{z_j z_k \cdot xw}}\right| \\ &= (\sigma_{z_i z_i \cdot xw} + \lambda_i^2) - \lambda_i^2.\end{aligned}$$

Therefore, the covariance structure of $\Sigma_{ss \cdot xw}$ can be described as $\Omega = \Sigma_{ss \cdot xw} + \boldsymbol{\lambda}\boldsymbol{\lambda}'$. Here, Ω is a positive diagonal matrix. Q.E.D.

4.3 DISCRETE CASE

In this section, we show that the similar result of Theorem 2 holds true in the case where all variables in a directed acyclic graph are dichotomous variables. For this purpose, based on the directed acyclic graph shown in Fig.2, suppose that $\mathrm{p}_{y|x=1,u=a,w=b} - \mathrm{p}_{y|x=0,u=a,w=b} = \tau_{yx}$ for any value $\boldsymbol{a} \cup \boldsymbol{b}$ in a set $\boldsymbol{U} \cup \boldsymbol{W}$, that is, τ_{yx} is constant regardless of $\boldsymbol{a} \cup \boldsymbol{b}$. Here, $\boldsymbol{U}$ represents a set of unobserved confounders, and $\boldsymbol{W}$ represents a set of observed covariates. In addition, let $\sigma_{xy \cdot w} = \mathrm{p}_{x=1,y=1|w=b} - \mathrm{p}_{x=1|w=b}\mathrm{p}_{y=1|w=b}$ for any $\boldsymbol{b}$. Similar notations are used for other parameters. Then, for Z_1, Z_2, Z_3, it is trivial that $\sigma_{z_i z_j \cdot w} = 0$ holds true from Fig.2. In addition,

$$\begin{aligned}&\mathrm{p}_{y=1|z_i=1,w=b} - \mathrm{p}_{y=1|z_i=0,w=b} \\ &= \sum_a (\mathrm{p}_{y=1|x=1,u=a,w=b} - \mathrm{p}_{y=1|x=0,u=a,w=b}) \\ &\quad \times (\mathrm{p}_{x=1|z_i=1,u=a,w=b} - \mathrm{p}_{x=1|z_i=0,u=a,w=b})\mathrm{p}_{u|w=b} \\ &= \tau_{yx}(\mathrm{p}_{x=1|z_i=1,w=b} - \mathrm{p}_{x=1|z_i=0,w=b}). \qquad (12)\end{aligned}$$

By using $\mathrm{p}_{x=1|w=b} = \mathrm{p}_{z_i=0|w=b}\mathrm{p}_{x=1|z_i=0,w=b} + \mathrm{p}_{z_i=1|w=b}\,\mathrm{p}_{x=1|z_i=1,w=b}$,

$$\begin{aligned}\sigma_{xz_i \cdot w} &= (\mathrm{p}_{x=1|z_i=1,w=b} - \mathrm{p}_{x=1|w=b})\mathrm{p}_{z_i=1|w=b} \\ &= (\mathrm{p}_{x=1|z_i=1,w=b} - \mathrm{p}_{x=1|z_i=0,w=b}) \\ &\quad \times \mathrm{p}_{z_i=0|w=b}\mathrm{p}_{z_i=1|w=b} \qquad (13)\end{aligned}$$

Furthermore, by using $\mathrm{p}_{y=1|w=b} = \mathrm{p}_{y=1|z_i=0,w=b} \times \mathrm{p}_{z_i=0|w=b} + \mathrm{p}_{y=1|z_i=1,w=b}\mathrm{p}_{z_i=1|w=b}$ and equations (12) and (13),

$$\begin{aligned}&\sigma_{yz_i \cdot w} \\ &= (\mathrm{p}_{y=1|z_i=1,w=b} - \mathrm{p}_{x|w=b})\mathrm{p}_{z_i=1|w=b} \\ &= (\mathrm{p}_{y|z_i=1,w=b} - \mathrm{p}_{y=1|z_i=0,w=b})\mathrm{p}_{z_i=0|w=b}\mathrm{p}_{z_i=1|w=b} \\ &= \tau_{yx}\sigma_{xz_i \cdot w}.\end{aligned}$$

Therefore, letting

$$\begin{aligned}\Sigma_{ss \cdot xw} &= \begin{pmatrix} \Sigma_{zz \cdot w} & \Sigma_{zy \cdot w} \\ \Sigma'_{zy \cdot w} & \sigma_{yy \cdot w} \end{pmatrix} \\ &\quad - \frac{1}{\sigma_{xx \cdot w}} \begin{pmatrix} \Sigma_{zx \cdot w} \\ \sigma_{yx \cdot w} \end{pmatrix} (\Sigma'_{zx \cdot w}, \sigma_{yx \cdot w}),\end{aligned}$$

we can obtain the same formulation as equation (8) by setting

$$\Omega = \begin{pmatrix} \Sigma_{zz \cdot w} & \mathbf{0} \\ \mathbf{0}' & \sigma_{yy \cdot w} - \frac{\sigma_{xy \cdot w}^2}{\sigma_{xx \cdot w}} + \sigma_{xx \cdot w}(\tau_{yx} - \frac{\sigma_{xy \cdot w}}{\sigma_{xx \cdot w}})^2 \end{pmatrix}$$

and

$$\boldsymbol{\lambda} = \begin{pmatrix} \frac{\Sigma_{zx \cdot w}}{\sqrt{\sigma_{xx \cdot w}}} \\ -\sqrt{\sigma_{xx \cdot w}}(\tau_{yx} - \frac{\sigma_{xy \cdot w}}{\sigma_{xx \cdot w}}) \end{pmatrix}.$$

Thus, based on the consideration above, it can be understood that the same result as Theorem 2 holds true in the case where all variables in a directed acyclic graph are dichotomous.

The instrumental inequality (Pearl, 1995) is well known as a necessary condition for testing whether a variable Z is an IV relative to (X, Y) in case where any element of $\boldsymbol{V}$ is discrete in a Bayesian network. This paper provides another testing criterion in discrete case in addition to Pearl's instrumental inequality.

5 APPLICATION

The above results are applicable to analyze the data obtained from a study on setting up painting conditions of car bodies, reported by Okuno et al. (1986). The data was collected with the purpose of setting up the process conditions, in order to increase transfer efficiency. The size of the sample is 38 and the variables of interest, each of which has zero mean and variance one, are the following:

Painting Condition : Dilution Ratio (X_1),
Degree of Viscosity (X_2), Painting Temperature (X_8)

Spraying Condition : Gun Speed (X_3),
Spray Distance (X_4), Atomizing Air Pressure (X_5),
Pattern Width (X_6), Fluid Output (X_7)

Environment Condition : Temperature (X_9),
Degree of Moisture (X_{10})

Response: Transfer Efficiency (Y)

Concerning this process, Kuroki et al. (2003) presented the directed acyclic graph shown in Fig.3.

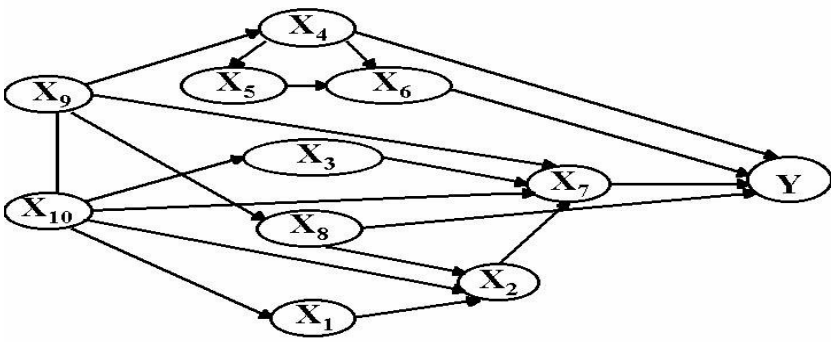


Fig.3 : Directed acyclic graph (3) (Kuroki et al., 2003)

Table 1 : The estimated correlation matrix based on Fig.3 (Kuroki et al., 2003)

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	Y
X_1	1.000	-0.736	-0.152	0.148	0.028	-0.042	0.324	0.216	0.283	-0.496	-0.091
X_2	-0.736	1.000	0.210	-0.331	-0.063	0.095	-0.479	-0.684	-0.635	0.684	0.326
X_3	-0.152	0.210	1.000	-0.091	-0.017	0.026	0.195	-0.134	-0.175	0.307	0.134
X_4	0.148	-0.331	-0.091	1.000	0.191	-0.286	0.184	0.397	0.521	-0.298	-0.614
X_5	0.028	-0.063	-0.017	0.191	1.000	0.291	0.035	0.076	0.099	-0.057	-0.277
X_6	-0.042	0.095	0.026	-0.286	0.291	1.000	-0.053	-0.114	-0.149	0.085	-0.250
X_7	0.324	-0.479	0.195	0.184	0.035	-0.053	1.000	0.396	0.353	-0.146	-0.044
X_8	0.216	-0.684	-0.134	0.397	0.076	-0.114	0.396	1.000	0.761	-0.435	-0.493
X_9	0.283	-0.635	-0.175	0.521	0.099	-0.149	0.353	0.761	1.000	-0.571	-0.475
X_{10}	-0.496	0.684	0.307	-0.298	-0.057	0.085	-0.146	-0.435	-0.571	1.000	0.283
Y	-0.091	0.326	0.134	-0.614	-0.277	-0.250	-0.044	-0.493	-0.475	0.283	1.000

Based on the directed acyclic graph, Kuroki et al. (2003) presented the estimated correlation matrix shown in Table 1.

In order to estimate the effect of X_7 on Y, since both X_1 and X_3 are conditional IVs given $\{X_9, X_{10}\}$, τ_{yx7} is identifiable through the observation of $\{X_1, X_7, X_9, X_{10}, Y\}$ or $\{X_3, X_7, X_9, X_{10}, Y\}$, and is given by $\tau_{yx7} = \sigma_{yx_i \cdot x_9 x_{10}} / \sigma_{x_7 x_i \cdot x_9 x_{10}} = 0.195$ $(i = 1, 3)$. In addition, $X_1 \perp\!\!\!\perp X_3 | \{X_9, X_{10}\}$ holds true.

Here, letting $\boldsymbol{S} = \{X_1, X_3, Y\}$, from Table 1, we can obtain

$$\Sigma_{ss \cdot x_7 x_9 x_{10}} = \begin{pmatrix} 0.682 & -0.069 & 0.0134 \\ -0.069 & 0.840 & 0.0130 \\ 0.0134 & 0.0130 & 0.756 \end{pmatrix}$$
$$= \begin{pmatrix} 0.754 & 0.000 & 0.000 \\ 0.000 & 0.906 & 0.000 \\ 0.000 & 0.000 & 0.756 \end{pmatrix}$$
$$- \begin{pmatrix} 0.268 \\ 0.257 \\ -0.051 \end{pmatrix} (0.268, 0.257, -0.051), \quad (14)$$

which is consistent with the form of equation (8). In addition, all the off-diagonal elements of $\Sigma_{ss \cdot x_7 x_9 x_{10}}$ are negative after sign changes of the third rows and the third columns. Furthermore, we can obtain $\sigma_{x_1 x_1 \cdot x_9 x_{10}} = 0.754$ and $\sigma_{x_3 x_3 \cdot x_9 x_{10}} = 0.906$ from Table 1, which are consistent with the (1,1) and the (2,2) components of diagonal matrix in equation (14), respectively. However, the (3,3) component is not equal to $\sigma_{yy \cdot x_9 x_{10}}$. This result indicates that both X_1 and X_3 can be regarded as conditional IVs given $\{X_9, X_{10}\}$ relative to (X_7, Y).

Here, noting that the numerator of equation (7) is not dependent on the selection of the IVs, we can obtain from Table 1,

$$\frac{a.var\left(\frac{\hat{\sigma}_{yx_1 \cdot x_9 x_{10}}}{\hat{\sigma}_{x_1 x_7 \cdot x_9 x_{10}}}\right)}{a.var\left(\frac{\hat{\sigma}_{yx_3 \cdot x_9 x_{10}}}{\hat{\sigma}_{x_3 x_7 \cdot x_9 x_{10}}}\right)} = \frac{\rho^2_{x_3 x_7 \cdot x_9 x_{10}}}{\rho^2_{x_1 x_7 \cdot x_9 x_{10}}} = \frac{0.270}{0.310} = 0.870,$$

which indicates that X_1 provides a smaller asymptotic variance of the total effect than X_3.

6 CONCLUSION

Instrumental variable method is a powerful tool to estimate total effects when unobserved variables are at presence. In order to identify covariates which satisfy the IV conditions, this paper proposed a method to test whether statistical data is generated by a model involving IVs. We showed that this method is applicable to both continuous case and discrete case. Thus, the results of this paper enable us apply the IV method to wider situations than before.

Finally, we would like to give some further topics. First, as we pointed out in section 1, our method may provide a tighter condition for IVs than Pearl's instrumental inequality. However, we did not provide the detailed discussion because of lack of space. In our opinion, a combination of Pearl's instrumental inequality and our criterion can provide tighter conditions than each of them. This requires further discussion. Second, when two or more covariates satisfy the IV conditions, two important problems occur: one is

the IV selection problem, and the other is the robust problem of causal claims through IVs (Pearl, 2004). Although the result of Kuroki and Cai (2004) can be used to solve the former problem, we leave the detailed discussion of applying our results to the latter problem for future research.

ACKNOWLEGDEMENT

The comments of the reviewers on preliminary versions of this paper are acknowledged. This research was supported by the Sumitomo Foundation, No.040705.

REFERENCES

Anderson, T. W. (1986). *Introduction to Multivariate Statistical Analysis*, Second edition. John Wiley & Sons.

Angrist, J. D., Imbens, G. W., and Rubin, D. B. (1996). Identification of Causal Effects using Instrumental Variables. *Journal of the American Statistical Association*,**91**, 444-472.

Balke, A. and Pearl, J. (1997). Bounds on Treatment Effects from Studies with Imperfect Compliance. *Journal of the American Statistical Association*,**92**, 1171-1176.

Bekker, P. A. and de Leeuw, J. (1987). The Rank of Reduced Dispersion Matrices. *Psychometrika*, **52**, 125-135.

Bollen, K. A. (1989), *Structural Equations with Latent Variables*, John Wiley & Sons.

Bollen, K. A., and Bauer, D. J. (2004). Automating the Selection of Model-Implied Instrumental Variables. *Sociological Methods and Research*, **32**, 425-452.

Bollen, K. A. and Ting, K. (2000). A Tetrad Test for Causal Indicators. *Psychological Methods*, **5**, 3-22.

Bowden, R. J. , and Turkington, D. A. (1984). *Instrumental Variables*, Cambridge University Press.

Brito, C. and Pearl, J. (2002). Generalized Instrumental Variables, *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, 85-93.

Chu, T., Scheines, R. and Spirtes, P. (2001). Semi-Instrumental Variables: A Test for Instrument Admissibility *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, 83-90.

Geiger, D., Verma, T. S. and Pearl, J. (1990). Identifying Independence in Bayesian Networks. *Networks*, **20**, 507-534.

Greenland, S. (2000). An Introduction to Instrumental Variables for Epidemiologists, *International Journal of Epidemiology*, **29**,722-729.

Kuroki, M. and Cai, Z. (2004). Selection of Identifiability Criteria for Total Effects by Using Path Diagrams. *Proceeding of the 20th Conference on Uncertainty in Artificial Intelligence*, 333-340.

Kuroki, M., Miyakawa, M. and Cai, Z. (2003). Joint Causal Effect in Linear Structural Equation Model and its Application to Process Analysis. *Proceeding of Ninth International Workshop on Artificial Intelligence and Statistics*, 70-77.

Lauritzen, S. L. (2004). Discussion on Causality, *Scandinavian Journal of Statistics*, **31**, 189-193.

Okuno, T., Katayama, Z., Kamigori, N., Itoh, T., Irikura, N. and Fujiwara, N. (1986). *Kougyou ni okeru Tahenryou Data no Kaiseki* (In Japanese), Nikkagiren, Tokyo.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Pearl, J. (1995). On the Testability of Causal Models with Latent and Instrumental Variables. *Proceeding of the 11th Conference on Uncertainty in Artificial Intelligence*, 435-443.

Pearl, J. (2000). *Causality: Models, Reasoning, and Inference*, Cambridge University Press.

Pearl, J. (2004). Robustness of Causal Claims, *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, 411-420.

Pearl, J. and Tarsi, M. (1986). Structuring Causal Trees, *Journal of Complexity*, **2**, 60-77.

Scheines, R., Cooper, G., Yoo, C. and Chu, T. (2001). Piecewise Linear Instrumental Variable Estimation of Causal Influence, *Proceeding of the 8th International Workshop on Artificial Intelligence and Statistics*, 286-291.

Spearman, C. (1904). General Intelligence Objectively Determined and Measured. *American Journal of Psychology*, **15**, 201-293.

Spearman, C. (1928). Pearson's Contribution to the Theory of Two Factors. *British Journal of Psychology*, **19**, 95-101.

Spirtes, P. , Glymour, C. , and Schienes, R. (1993). *Causation, Prediction, and Search*. Springer-Verlag.

Xu, L. and Pearl, J. (1989). Structuring Causal Tree Models with Continuous Variables, *Uncertainty in Artificial Intelligence*, **3**, 209-219.

Estimating Class Membership Probabilities using Classifier Learners

John Langford
Toyota Technological Institute at Chicago
Chicago, IL 60637
jl@tti-c.org

Bianca Zadrozny
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
zadrozny@us.ibm.com

Abstract

We present an algorithm, "Probing", which reduces learning an estimator of class probability membership to learning binary classifiers. The reduction comes with a theoretical guarantee: a small error rate for binary classification implies accurate estimation of class membership probabilities. We tested our reduction on several datasets with several classifier learning algorithms. The results show strong performance as compared to other common methods for obtaining class membership probability estimates from classifiers.

1 Introduction

Background. Classifier learning is the problem of inducing a predictor for distinguishing between two (or a few) classes given a set of labeled training examples. There are many reasons for considering classifier learning as the most fundamental learning task. Predicting one bit (or a few bits) is as simple as non-trivial prediction can be. Because of this simplicity, classifier learning is often more amenable to study than other learning problems. On the practical side, several fast learning algorithms (such as SVM [13], boosting [6] on decision trees [14], and neural networks [9]) exist which generally lead to classifiers with excellent predictive performance.

The Problem. On the other hand, there are several real-world applications requiring probabilistic answers instead of discrete class labels. For example, in medical domains, doctors often prefer to make decisions based upon probabilistic predictions of the patient state, rather than being given a discrete chosen action for each patient. Even when the actual probability estimates are not required, a ranking of examples from the most likely member to the least likely member of a class may be desirable. This is the case, for example, in document categorization, where users might like to see a list of documents ranked by their relevance to a particular topic.

Probability estimates are also important when the classification outputs are not used in isolation but are combined with information from other components in a system. For example, in handwritten character recognition the outputs from the classifier are used as input to a high-level system which incorporates domain information, such as a language model. In this case, probabilities provide a standard language for summarizing information from multiple sources into a single decision. When the components of the system are distributed, bandwidth constraints may make this summarization necessary as well as standard.

Solution Sketch. Our method for answering the needs of these real-world applications can be thought of as a reduction of class probability estimation to classifier learning. The reduction (which we name "Probing") satisfies strong optimality guarantees: good performance with respect to classification implies good performance with respect to class probability estimation. The essential observation underlying the Probing reduction is that probability estimates (in a Bayesian sense) can be extracted from preferences over bets at different odds ratios. The machinery that applies this observation to classifier learning algorithms is the Probing reduction.

2 Related Work

There are several existing approaches for obtaining class probability estimates from classifiers.

Bagging. One common approach uses bagging [12]. Essentially, the learning algorithm is run on many training sets formed by sampling with replacement from the original dataset, and the estimated probability that any sample x has label $y = 1$ is the proportion of learned classifiers that output 1 for x. The results of the bagging approach may have nothing to do with $\Pr_{(x,y)\sim D}(y = 1|x)$, the probability accord-

ing to the data generation process that $y = 1$. As has been observed before, the probability estimates obtained through bagging are measuring the instability of the classifier learning method over data perturbations [10] rather than $\Pr_{(x,y)\sim D}(y = 1|x)$. Indeed, for larger datasets, bagging with certain learning algorithms might result in the same classifier on every run, causing the bagging probability to be always 0 or 1 even when the fundamental process is much less certain. In contrast, our Probing reduction has a simple guarantee—if the classifiers solve their associated classification problems optimally, then the reported probability is precisely $\Pr_{(x,y)\sim D}(y = 1|x)$. In addition, unlike bagging, Probing preserves independence in the original dataset[1], which is quite important for many learning algorithms (see [17] for further discussion).

Margin Sigmoid. Another approach for extracting probabilities from classifiers is to take an internal representation such as the margin of a support vector machine and transform it into a value $v \in [0, 1]$ to achieve calibration on the training set (for example, using a sigmoid function [11]). The Probing reduction is more general (since it applies to classifiers without an internal margin such as decision trees) and less adhoc. In particular, in the Probing reduction a small classification error rate necessarily implies good probability estimates over all distributions generating the data. No guarantee of this quality can be made for any transformation of a margin into a $[0, 1]$ interval, essentially because the margin is not a "sufficient statistic" for probability estimation. See also [2] for a discussion of sparsity versus probability estimation issues.

Direct Prediction. The third standard approach is to simply predict probabilities directly, tipically with some form of learned probabilistic model. Our results can be viewed from this approach as (1) providing compatibility between non-probabilistic and probabilistic models and (2) enriching the set of probabilistic models with new and often better performing models.

We discovered that a reduction similar to Probing had been published previously by Halck [7]. The main difference is that Probing reweights the classes appropriately to obtain preferences over bets at different odds ratios (see next section), while Halck's reduction changes the labels of examples randomly for the same purpose. We provide a theoretical performance guarantee and present a comprehensive experimental evaluation of Probing. Halck's method may be analyzable and may have good empirical performance, but the theory and evaluation have not been done yet.

[1]Resampling with replacement provides samples from the correct distribution in a *dependent* manner.

3 The Probing Reduction

Assume we have examples (x, y) drawn from an unknown distribution D with domain $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is the feature space and $\mathcal{Y} = \{0, 1\}$ is the label space.

Basic Observations. The Probing reduction is based upon the observation that a binary classifier which predicts 1 for an example x is implicitly predicting that the probability of class 1 given x is greater than the probability of class 0 given x, that is, $D(y = 1|x) > D(y = 0|x)$. Conversely, a classifier which predicts 0 for x suggests that $D(y = 0|x) > D(y = 1|x)$.

By weighting the training examples such that positive examples are w times more costly to classify incorrectly than negative examples, it is possible to learn classifiers which predict 1 if

$$D(y = 1|x) > \frac{1}{1 + w}$$

and 0 otherwise.

Corollary 1 *(Minimal Error Probing) Let* $i_p(y) = \frac{1-p}{p}y + (1 - y)$. *Then*

$$\arg \min_{c:X\to Y} E_{(x,y)\sim D}\ i_p(y)\ I(c(x) \neq y)$$

$$= I\left(D(y = 1|x) > p\right)$$

except on a set of measure 0.

This corollary only discusses *minimal* error classifiers, so it only applies in the asymptotic limit of a bayes consistent classifier. However, the probing algorithm can be proved sound even for classifiers with nonminimal error rates. We prove this more general theorem in section 4 (theorem 2). The corollary states that a classifier which minimizes a importance weighted loss can be used to determine if the conditional distribution $D(y = 1|x)$ is above or below a threshold for a given value of x, where the threshold depends on the relative weight given to positive and negative examples.

The Algorithm. Now, suppose that we learn a classifier c_p for every choice of p and each of these classifiers has a minimal loss. Then, for an input x, each of the classifiers answers whether $D(y = 1|x)$ is greater than the threshold p. For $p = 0$, we necessarily have $c_p(x) = 1$ because $D(y = 1|x)$ cannot be less than 0. As p grows, the predictions of the subsequent classifiers are monotone in p with one point $p^* = D(y = 1|x)$ where the prediction changes from 1 to 0. Therefore, the Probing reduction reports:

$$D(y = 1|x) = p^*. \tag{1}$$

Three issues remain:

1. Not all classifier learners automatically take the weight w as input. We address this by using the

Algorithm 1 Probing-Learner (classifier learner A, dataset $S = (x, y)^m$, number of iterations t)

1. Let $I = \{[0, 1]\}$
2. For $i = 1$ to t
   (a) Let $[a, b]$ = minimax interval $\in I$
   (b) Let p = minimax optimal $p \in [a, b]$
   (c) Let $I = (I - \{[a, b]\}) \cup \{[a, p], [p, b]\}$
   (d) Let $S_p = \{(x, y, i_p(y)) : (x, y) \in S\}$
   (e) Let $c_p = A(S_p)$
3. Return all c_p

Algorithm 2 Probing-Predictor (set of weighted classifiers c_p, input x)

1. Let $n = \sum_p c_p(x)$
2. Let $[a, b]$ = the nth interval
3. Return minimax optimal $p \in [a, b]$.

costing reduction [17] from importance weighted classification to binary classification, which applies to any classifier learner.

2. Learning a classifier for every choice of w is too computationally intense. We resolve this by discretizing the set of w and reporting a probability in the discrete interval.
3. Independently learned sub-optimal classifiers may not be consistent, meaning that the sequence of predictions for a given example is not necessarily monotonic. We simply sort the results of the classifiers to make the sequence monotonic (all zeros before all ones). This gives us the solution consistent with the largest possible number of classifiers. Although this may appear to be a hack, the proof of theorem 2 fundamentally relies upon the sort.

Learning. The algorithms for learning and making predictions using the Probing reduction are reported in figures Algorithm 1 and Algorithm 2. The Probing-Learner algorithm takes as input a classifier learning algorithm, a set of training examples and some number of iterations. On each iteration, it refines the discretization of the interval with the largest potential gain from refinement (this is loss dependent, see below for details). For each chosen interval a minimax choice of probability and the corresponding weight (obtained by inverting equation 1) are calculated and a weighted classifier is learned.

Prediction. The Probing-Predictor algorithm takes as input the set of weighted classifiers and an input x. It sums the predictions of each classifier on x to determine the number of classifiers that predict 1 for x. It then picks the minimax probability in the nth interval (counting from 0). This is equivalent to sorting the classifier predictions to obtain monotonicity and reporting a probability from the interval in which the prediction changes from 1 to 0.

Loss Details. Steps 2(a) and 2(b) in Probing-Learner are dependent on the probabilistic loss function of interest. Here we describe these steps for two of the most commonly used probabilistic loss functions: cross entropy and squared error. In order to determine the minimax interval and the minimax probability for these loss functions, we make the somewhat unjustified but convenient assumption that all classifiers are correct.

1. Cross entropy is given by

$$E_{(x,y)\sim D} I(y = 1) \ln \frac{1}{\hat{p}(x)} + I(y = 0) \ln \frac{1}{1 - \hat{p}(x)}.$$

The minimax optimal $\hat{p} \in [a, b]$ is given by:

$$\begin{aligned} &\min_{\hat{p}\in[a,b]} \max_{p\in[a,b]} [\text{loss on } \hat{p} - \text{loss on } p] \\ &= \min_{\hat{p}\in[a,b]} \max_{p\in[a,b]} \left[p \ln \tfrac{1}{\hat{p}} + (1-p) \ln \tfrac{1}{1-\hat{p}} \right. \\ &\qquad \left. -p \ln \tfrac{1}{p} - (1-p) \ln \tfrac{1}{1-p} \right] \end{aligned}$$

This implies that

$$\hat{p} = \frac{1}{1 + e^{\frac{H(b) - H(a)}{b-a}}},$$

where $H(a)$ is Shannon entropy (in nats) of a coin with bias a.

The minimax interval is the interval which creates the largest potential change in loss when refined on the training set. Suppose we let S_a be the set of training examples whose current probabilistic prediction is $\hat{p} \in [a, b]$. The chosen [2] interval is:

$$\arg \max_{[a,b]\in I} |S_a| \left[(1-a) \ln \frac{1-a}{1-\hat{p}} + a \ln \frac{a}{\hat{p}} \right]$$

2. Mean squared error is given by

$$E_{(x,y)\sim D}[(y - D(y|x))^2].$$

The minimax optimal $\hat{p} \in [a, b]$ is given by:

$$\begin{aligned} &\min_{\hat{p}\in[a,b]} \max_{p\in[a,b]} [\text{loss on } \hat{p} - \text{loss on } p] \\ &= \min_{\hat{p}\in[a,b]} \max_{p\in[a,b]} \left[p(1-\hat{p})^2 + (1-p)\hat{p}^2 \right. \\ &\qquad \left. -p(1-p)^2 - (1-p)p^2 \right] \end{aligned}$$

This implies that

$$\hat{p} = (a + b)/2.$$

[2] Note that the definition $\hat{p}$ implies this is equivalent to $\arg\max_{[a,b]\in I} |S_a| \left[(1-b) \ln \frac{1-b}{1-\hat{p}} + b \ln \frac{b}{\hat{p}} \right]$

The minimax interval is similarly given by

$$\arg \max_{[a,b] \in I} |S_a|(b-a).$$

In a transductive setting, using the test set distribution to weight the minimax interval is a better than $|S_a|$.

Multiclass Case. For the multiclass case with $k > 2$ labels, we simply apply the binary method to $k-1$ binary problems defined by the mapping:

$$(x, y) \to (x, I(l = y))$$

for $k-1$ choices of l. This method gives us probability estimates $\hat{p}_1, ..., \hat{p}_{k-1}$ for $k-1$ classes, so we can estimate the probability of the last class according to $\hat{p}_k = 1 - \sum_{i=1}^{k-1} \hat{p}_i$.

4 Analysis of the Probing Reduction

The following theorem shows that the probing reduction is inherently robust against classification errors. For the theorem, remember $i_p(y) = \frac{1-p}{p} y + (1-y)$ and let the importance weighted loss of a classifier c under distribution D be

$$l_p(c) = \frac{1}{Z_p} E_{(x,y) \sim D} i_p(y) I(c(x) \neq y)$$

where $Z_p = E_{(x,y) \sim D'} \; i_p(y)$ is a normalization constant.

This theorem analyzes probing in the limit as the number of classifiers approaches ∞ uniformly over the unit interval. Discretization to intervals of size d adds at most $d + d^2/4$ to the expected squared error.

Theorem 2 (Probing Error Transformation) If the classifiers c_p learned under Probing have average relative importance weighted loss

$$\epsilon = E_{p \sim U(0,1)} \left[l_p(c_p) - \min_c l_p(c) \right],$$

then

$$E_{x \sim D}(D(1|x) - \hat{p}(x))^2 \leq \; 2 \max\{D(1), D(0)\} \epsilon \leq 2\epsilon$$

where $\hat{p}(x)$ is the probing prediction.

This is a strong theorem in the sense that it ties the error in the probability predictions to the *average relative* importance weighted loss of the classifiers. Using the average loss over w results in a more powerful statement than using the maximal loss, because it is easier to obtain a set of classifiers which have small average loss than to obtain a set of classifiers, all with a small loss. Using a loss that is relative to the loss of the Bayes optimal classifier means that the theorem applies even when the fundamental noise rate is large.

Note that when the proportion of positive and negative examples is balanced (D(1)=D(0)=0.5) the bound on the error in the probability predictions is exactly the average relative importance weighted loss,

$$E_{x \sim D}(D(1|x) - \hat{p}(x))^2 \leq \epsilon$$

Proof: Let $c_p^* = \min_c l_p(c)$, then

$$\begin{aligned}
\epsilon &= E_{p \sim U(0,1)} \left[l_p(c_p) - l_p(c_p^*) \right] \\
&= E_p \left[\tfrac{1}{Z_p} E_{(x,y) \sim D} [i_p(y)(I(c_p(x) \neq y) - I(c_p^*(x) \neq y))] \right] \\
&= E_p \left[\tfrac{1}{Z_p} E_x [D(1|x) \tfrac{1-p}{p} (c_p^*(x) - c_p(x)) \right. \\
&\qquad \left. + D(0|x)(c_p(x) - c_p^*(x))] \right] \\
&= E_p \left[\tfrac{1}{Z_p} E_x [D(1|x) \tfrac{1-p}{p} (c_p^*(x) - c_p(x)) \right. \\
&\qquad \left. + (1 - D(1|x))(c_p(x) - c_p^*(x))] \right] \\
&= E_p \left[\tfrac{1}{Z_p} E_x [(c_p^*(x) - c_p(x))(\tfrac{1}{p} D(1|x) - 1)] \right].
\end{aligned}$$

Rewriting with $D(1|x) > p \Rightarrow c_p^*(x) = 1$ and $D(1|x) < p \Rightarrow c_p^*(x) = 0$, we have

$$\epsilon = E_p \left[\frac{1}{p Z_p} E_x [I(c_p^*(x) \neq c_p(x)) \, |D(1|x) - p|] \right].$$

Notice that for all p, $Z_p = D(0) + D(1)\frac{1-p}{p}$. Hence

$$pZ_p = pD(0) + (1-p)D(1) \leq \max\{D(1), D(0)\}.$$

Therefore, we have

$$E_p E_x [I(c_p^* \neq c_p(x)) | D(1|x) - p|] \leq \max\{D(1), D(0)\} \epsilon,$$

which can be re-written as

$$\begin{aligned}
E_{x \sim D} \int_0^1 I(c_p^* \neq c_p(x)) |D(1|x) - p| d_p \\
\leq \max\{D(1), D(0)\} \epsilon, \qquad (2)
\end{aligned}$$

Given a fixed budget of ϵ for binary errors, the scenario which maximizes probability error estimates has all classifiers erring in one direction (either for $p < D(1|x)$ or $p > D(1|x)$) because two errors in different directions are canceled in the sorting phase of Probing-Predictor. Furthermore, errors by classifiers nearer to $D(1|x)$ are preferred to errors far from $D(1|x)$, since the importance weighted loss payed by the adversary increases monotonically with distance from $D(1|x)$. Therefore, when the Probing prediction is $\hat{p}(x)$, all the classifiers c_p for which the corresponding p are between $\hat{p}(x)$ and the correct probability $D(1|x)$ give incorrect predictions whereas the other classifiers give correct predictions. As a result, the integrand of equation 2 is non-zero only for $\hat{p}(x) < p < D(1|x)$ (if the classifiers err in one direction) or $D(1|x) < p < \hat{p}(x)$ (if the classifiers err in the other direction). Thus, we can re-write equation 2 as

$$E_{x \sim D} \left| \int_{\hat{p}(x)}^{D(1|x)} |D(1|x) - p| d_p \right| \leq \max\{D(1), D(0)\} \epsilon.$$

By solving the integral, we get

$E_{x\sim D}(D(1|x) - \hat{p}(x))^2 \leq 2\max\{D(1), D(0)\}\epsilon$. ∎

In this theorem, we measure the probabilistic loss using the true distribution $D(1|x)$. Because this distribution is unknown for most real-world problems, it is important to understand the implications of the theorem to metrics that only use the class labels. We consider these implications for three of these metrics in the following subsections.

4.1 The squared error metric for classification

The loss in the squared error metric for classification is defined as

$$E_{x,y\sim D}(y - \hat{p}(x))^2.$$

Note that

$$\begin{aligned}
&E_{x,y\sim D}(y - D(1|x))^2 + (\hat{p}(x) - D(1|x))^2 \\
&= E_{x\sim D}D(1|x)[(1 - D(1|x))^2 + (\hat{p}(x) - D(1|x))^2 \\
&\qquad +(1 - D(1|x))[D(1|x)^2 + (\hat{p}(x) - D(1|x))^2] \\
&= E_x D(1|x)[1 - 2D(1|x)] + D(1|x)^2 + (\hat{p}(x) - D(1|x))^2 \\
&= E_x D(1|x) + \hat{p}(x)^2 - 2\hat{p}(x)D(1|x) \\
&= E_x(\hat{p}(x) - D(1|x))^2 + D(1|x) - D(1|x)^2 \\
&= E_x(\hat{p}(x) - D(1|x))^2 + C
\end{aligned}$$

where C is some problem dependent constant.

Consequently, optimizing the squared error metric for classification implicitly optimizes the squared error in the probability estimate.

4.2 The cross entropy metric for classification

The loss in the cross entropy metric is $E_{x,y\sim D}I(y = 1)\ln\frac{1}{\hat{p}(x)} + I(y = 0)\ln\frac{1}{1-\hat{p}(x)}$. Cross entropy differs fundamentally from the squared error metric because it is (theoretically) unbounded. Consequently, the optimal strategy of an adversarial binary classifier is to err simultaneously with all classifiers on one side of $D(1|x)$, producing an unbounded loss when the discretization goes to 0. For cross entropy we can only prove a much weaker theorem: minimal error implies $\hat{p}(x)$ is optimal up to the discretization.

In practice, this worst case behavior can be avoided. The asymptote to ∞ is very slow as $\hat{p}(x)$ approaches 0 or 1. Consequently, we can project $\hat{p}(x)$ into the nearest element in the interval $[0.001, 0.999]$ and incur a maximal loss of about $\log(1/0.001)\epsilon + 0.001 = 6.9\epsilon + 0.001$ (when $D(1) = D(0)$), while preserving almost all of the dynamic range.

4.3 Implications for ordering metrics

Relative Ordering. In some situations, we may be interested only in the relative ordering or ranking of examples given by the class probability estimates. Unfortunately, very small errors in the ordering with respect to $[0, 1]$ can result in very large misorderings with respect to a subset of $[0, 1]$ In particular, suppose that the optimal answer is $\hat{p}(x) = 0.5$, but the answer we provide is $\hat{p}(x) = 0.49999$. If every other element in the ordering is in the interval $(0.49999, 0.5)$, then this small error results in a very large ordering error with respect to the set of elements.

AUC analysis. Area under the ROC curve (AUC) is one commonly used relative ordering metric. Suppose we have one example with label $y = 1$, $D(y = 1|x) = 1$ and Probing estimates $\hat{P}(y = 1|x) = 0$, and n examples with label $y = 0$, $D(Y = 1|x) = 1/n$ where Probing estimates $\hat{P}(y = 1|x) = 1/n$. In this setting, the AUC (as defined in [1]) is 0. Despite this terrible performance, the classifiers need err only once. This is less than any fixed importance weighted loss rate for sufficiently large n.

5 Experimental Results

Here we present the results obtained by applying the Probing reduction to fifteen (binary) datasets: eleven from the UCI machine learning repository [5] (Adult, Australian Credit, Breast, Diabetes, Echocardiogram, Hepatitis, Ionosphere, Kr-vs-kp, Liver, Mushroom, Sick), two from the UCI KDD archive [8] (KDD Cup 98 and COIL) and two from the 2004 KDD Cup (Biology and Physics).

We use four different base classifier learners available within the machine learning tool Weka [15]: the J48 decision tree learner, the SMO support vector machine (SVM) learner (linear kernel), Naive Bayes and logistic regression. For choosing the weights, we use the Probing variant that attempts to optimize cross entropy for 100 iterations. For realizing the importance weighted classification, we use rejection sampling (proportionally to the weights) for the decision tree learner and for logistic regression, and give the weights directly to the learners for SVM and Naive Bayes (see [17]).

We are interested in comparing the performance of the Probing reduction with standard methods for obtaining probability estimates from these classifiers. Decision trees and Naive Bayes produce an internal score in the interval $[0, 1]$ that can be used as a class probability estimate, but they are known to be inaccurate [16]. For this reason, besides testing their performance, we have also tested well-known methods for obtaining better probability estimates from these classifiers: bagging for decision trees (100 classifiers) and calibration using a sigmoid function for Naive Bayes [3]. Since the SVM classifier does not produce probability estimates, we have tested only the standard method for obtaining probabilities from SVM (also available in Weka): calibration using a sigmoid function [11].

For datasets that have a standard training/test split in the UCI repository, we use the standard split. For the other datasets, we randomly split the data into a training set with 66% of the examples and a test set with 33% of the examples.

We use three different metrics for assessing the accuracy of the probability estimates: root mean squared error (RMS), cross entropy (CXE) and area under the ROC curve (AUC). For more details about these metrics and an analysis of their characteristics, see [4]. The results on the test set using CXE, RMS and AUC are shown, respectively, in tables 1, 2 and 3. Note that for RMS and CXE the smaller the value the better the performance, while for AUC the larger the value the better the performance. Note also that AUC only measures how well the probability estimates rank the examples from the most probable to the least probable member of the class, while RMS and CXE also measure the calibration of the estimates.

Decision tree results. The Probing reduction outperforms a single J48 classifier for all datasets but Mushroom and for the majority of the datasets also outperforms Bagging J48, on the three metrics. A cross entropy of ∞ means that the classifier predicted probability one for an example whose label is zero (or vice-versa). This is often the case for the J48 classifier, showing that its internal scores are not reasonable class probability estimates for cross entropy loss. Bagging solves this problem by taking advantage of the instability of the learner over different training sets to average out extreme results. However, in cases where there is no instability (like the KDD Cup 98 dataset) bagging performs the same as the base classifiers. The same would be true if we apply bagging to other learners that are stable such as SVM and Naive Bayes.

SVM Results. The Probing reduction outperforms the SVM with sigmoid calibration for most datasets on all three metrics. The AUC metric results show a greater advantage of Probing over sigmoid calibration than the two other metrics. This is probably due to the fact that the sigmoid calibration maintains the same ranking as given by the SVM margins while Probing can give a better ranking based on the results of the different weighted classifiers.

Naive Bayes results. The Probing reduction outperforms a single Naive Bayes classifier for most datasets on the cross entropy and squared error metrics. On the AUC metric, however, a single Naive Bayes classifier usually performs slightly better than Probing Naive Bayes. This may be due to Naive Bayes estimates ranking well (although with terrible probability estimates), while Probing estimates have less resolution leading to more ties. The results also show that Naive Bayes with sigmoid calibration outperforms Probing Naive Bayes on most datasets in all three metrics. Naive Bayes is not a very accurate classifiers so the Probing reduction, which relies on that accuracy for different weight settings, is not optimal. On the other hand, the sigmoid approach has a separate fitting stage which can repair bad estimates to some extent. Furthermore, the sigmoid transformation maintains approximately the same ranking of examples as the original Naive Bayes classifier, leading to approximately the same AUC.

Dataset	RMS	CXE	AUC
Adult	ProbJ48	ProbJ48	ProbJ48
Australian	ProbJ48	ProbJ48	ProbJ48
Biology	SigSVM	SigSVM	Prob(J48/Log)
Breast	SigNB	SigNB	ProbNB
COIL	ProbJ48	ProbJ48	ProbJ48
Diabetes	Log	Log	ProbSVM
Echocardio	SigNB	SigNB	Log
Hepatitis	BagJ48	ProbSVM	ProbNB
Ionosphere	BagJ48	BagJ48	BagJ48
KDD-98	ProbJ48	ProbJ48	ProbLog
Kr-vs-kp	BagJ48	BagJ48	BagJ48
Liver	BagJ48	BagJ48	BagJ48
Mushroom	J48/SVM	J48/SVM	J48/SVM
Physics	ProbJ48	ProbJ48	ProbJ48
Sick	BagJ48	BagJ48	ProbJ48

Table 4: Best method per dataset/metric.

Logistic Regression. The Probing reduction performs better than a single Logistic Regression classifier for optimizing cross-entropy. In some cases, Logistic Regression yields a cross entropy of ∞, while Probing gives a reasonable result. For the two other metrics there is not a clear difference.

To get a better idea of which algorithms are performing best, we have one last table consisting of the method that led to best test performance for each metric and dataset in table 4. Probing wins in 20 out of 45 experiments against the other tested alternatives (J48, Naive Bayes, Sigmoid Naive Bayes, Sigmoid SVM, Bagged Decision Tree and Logistic Regression). It is also interesting to note that the performance of the decision tree, J48, is very good. This may be due[3] to the fact that the other learners that we are using are linear (we have not tried different kernels for SVM).

Finally, to give an idea of the speed of convergence of Probing in practice, we plot the cross entropy performance (using J48) against the number of Probing iterations for 5 representative datasets in figure 1. Probing seems to converge to a reasonable probability estimate after 20 or so iterations.

6 Discussion

Probing is a general technique for converting any classifier learner into a class probability estimator.

[3]It's also possible that the design of J48 (which inherits from C4.5) is overfit to the UCI datasets.

Dataset	J48	BagJ48	ProbJ48	SigSVM	ProbSVM	NB	SigNB	ProbNB	Log	ProbLog
Adult	∞	0.499	**0.436**	**0.472**	0.479	0.949	0.521	**0.464**	∞	**0.460**
Australian	∞	0.444	**0.442**	0.487	**0.471**	∞	**0.637**	0.656	0.531	0.503
Biology	∞	0.018	**0.017**	**0.016**	0.021	∞	0.071	**0.057**	0.018	**0.016**
Breast	∞	0.793	**0.791**	0.837	**0.809**	0.840	**0.753**	0.830	1.07	**0.850**
COIL	0.326	0.323	**0.301**	0.325	**0.311**	0.357	**0.305**	0.350	0.304	**0.303**
Diabetes	∞	0.806	**0.784**	**0.758**	0.775	∞	0.861	**0.780**	**0.746**	0.774
Echocardio	1.216	0.924	**0.897**	0.891	**0.837**	0.902	**0.810**	0.852	**0.829**	0.847
Hepatitis	0.733	**0.655**	0.665	0.695	**0.660**	1.416	**0.733**	0.953	2.45	**1.21**
Ionosphere	∞	**0.210**	0.287	0.418	**0.416**	∞	0.353	**0.304**	0.303	**0.297**
KDD-98	0.289	0.289	**0.283**	**0.285**	**0.285**	∞	**0.285**	0.351	**0.283**	**0.283**
Kr-vs-kp	∞	**0.026**	0.069	0.173	**0.128**	0.416	**0.406**	0.418	0.145	**0.103**
Liver	∞	**0.832**	0.888	**0.885**	0.891	0.940	**0.917**	0.950	**0.882**	0.892
Mushroom	**0.000**	**0.000**	0.006	**0.000**	0.001	0.191	**0.166**	0.298	**0.000**	0.003
Physics	∞	0.763	**0.762**	**0.802**	0.805	∞	**0.880**	0.933	0.792	**0.783**
Sick	∞	**0.070**	0.086	**0.171**	0.203	0.219	0.190	**0.183**	∞	0.164

Table 1: Cross entropy results. The best results for each classifier (J48, SVM and Naive Bayes) are in bold.

Dataset	J48	BagJ48	ProbJ48	SigSVM	ProbSVM	NB	SigNB	ProbNB	Log	ProbLog
Adult	0.347	0.331	**0.310**	**0.324**	0.326	0.335	0.332	**0.320**	0.321	**0.320**
Australian	0.328	0.303	**0.298**	0.320	**0.313**	0.382	**0.358**	0.363	**0.321**	0.322
Biology	0.061	**0.048**	0.048	**0.047**	0.052	0.075	0.094	**0.057**	0.050	**0.049**
Breast	0.439	**0.426**	**0.426**	0.439	**0.436**	0.433	**0.415**	0.432	0.457	**0.448**
COIL	0.237	0.236	**0.232**	0.236	**0.234**	0.254	**0.233**	0.254	**0.233**	**0.233**
Diabetes	0.474	0.429	**0.428**	**0.412**	0.424	0.460	0.450	**0.429**	**0.410**	0.416
Echocardio	0.539	0.470	**0.466**	0.464	**0.447**	0.445	**0.435**	0.449	**0.448**	0.454
Hepatitis	0.390	0.383	**0.368**	0.368	**0.358**	0.432	**0.400**	0.404	**0.440**	0.449
Ionosphere	0.303	**0.201**	0.215	0.295	**0.284**	0.259	0.251	**0.238**	0.304	**0.297**
KDD-98	0.219	0.219	**0.218**	0.219	**0.218**	0.233	**0.219**	0.247	**0.218**	**0.218**
Kr-vs-kp	0.093	**0.065**	0.091	0.179	**0.160**	0.300	**0.295**	0.300	0.158	**0.142**
Liver	0.560	**0.439**	0.461	**0.459**	0.461	0.473	**0.461**	0.479	0.457	**0.456**
Mushroom	**0.000**	0.001	0.012	**0.000**	0.003	0.182	**0.170**	0.201	**0.000**	0.008
Physics	0.529	0.425	**0.424**	0.435	**0.434**	0.490	0.490	**0.477**	0.432	**0.430**
Sick	0.125	**0.116**	0.119	**0.175**	0.205	0.199	**0.187**	0.190	0.175	**0.172**

Table 2: Squared error results. The best results for each classifier (J48, SVM and Naive Bayes) are in bold.

Dataset	J48	BagJ48	ProbJ48	SigSVM	ProbSVM	NB	SigNB	ProbNB	Log	ProbLog
Adult	0.852	0.891	**0.912**	0.901	**0.901**	**0.906**	**0.906**	0.905	0.902	**0.904**
Australian	0.900	0.937	**0.944**	0.912	**0.924**	**0.920**	**0.920**	0.917	0.918	**0.926**
Biology	0.885	0.976	**0.987**	**0.986**	0.982	**0.957**	0.922	0.956	0.981	**0.987**
Breast	0.606	0.602	**0.633**	**0.680**	0.659	0.723	0.723	**0.724**	**0.642**	0.611
COIL	0.500	0.641	**0.720**	0.542	**0.682**	0.709	0.709	**0.717**	0.706	**0.708**
Diabetes	0.715	0.796	**0.798**	0.817	**0.824**	0.779	0.779	**0.805**	**0.820**	0.819
Echocardio	0.543	0.644	**0.645**	0.704	**0.743**	**0.759**	**0.759**	0.753	**0.770**	0.755
Hepatitis	0.654	**0.670**	0.662	**0.749**	0.748	0.743	0.743	**0.757**	0.629	**0.667**
Ionosphere	0.895	**0.981**	0.966	0.926	**0.928**	0.960	**0.960**	0.957	0.905	**0.952**
KDD-98	0.500	0.500	**0.616**	0.598	**0.602**	**0.606**	**0.606**	0.604	0.616	**0.617**
Kr-vs-kp	0.998	**1.000**	0.999	0.992	**0.995**	**0.953**	**0.953**	0.953	0.995	**0.998**
Liver	0.607	**0.754**	0.724	0.711	**0.714**	**0.713**	**0.713**	0.706	0.710	**0.711**
Mushroom	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.998**	**0.998**	0.990	**1.000**	**1.000**
Physics	0.674	0.803	**0.803**	0.785	**0.789**	**0.738**	**0.738**	0.729	0.788	**0.791**
Sick	0.965	0.985	**0.989**	**0.946**	0.940	0.949	0.949	**0.952**	0.935	**0.952**

Table 3: Area under the ROC results. The best results for each classifier (J48, SVM and Naive Bayes) are in bold.

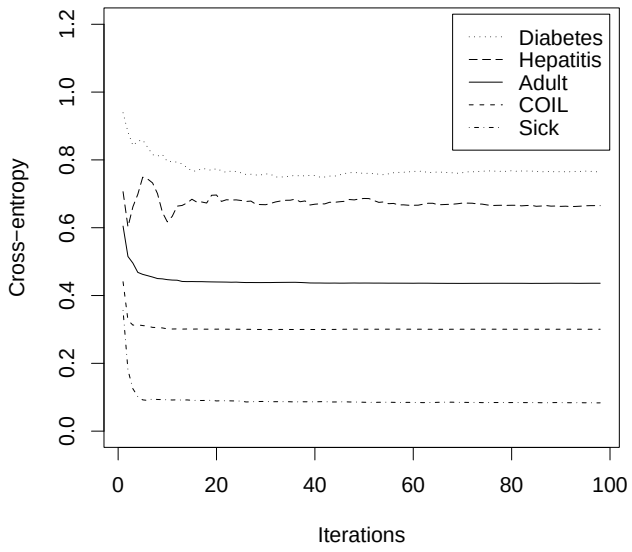


Figure 1: Cross entropy vs. the number of Probing iterations for 5 representative datasets (using J48).

Theorem 2 shows that the probing reduction is well founded: good performance on the created classification problems implies good performance with respect to Probing probability estimates. Experimental results show that Probing achieves strong performance relative to a variety of other (typically more specialized) techniques using a variety of classifier learners. This combination of positive results is unmatched by any alternative method.

In future work, we would like to validate the method on multiclass problems as outlined in section 3. Also, further studying the behavior of different discretization schemes for choosing the weights could lead to enhancements of the Probing estimates for different performance criteria.

References

[1] S. Agarwal, T. Graepel, R. Herbrich, S. Har-Peled and D. Roth (2004). Generalization bounds for the Area under an ROC curve. Technical report UIUCDCS-R-2004-2433, Computer Science Department, University of Illinois at Urbana-Champaign.

[2] P. Bartlett and A. Tewari (2004). Sparseness versus estimating conditional probabilities: Some asymptotic results. In *Proceedings of the 17th Annual Conference on Learning Theory* (COLT-2004), 564-578. Springer-Verlag.

[3] P. Bennett (2000). Assessing the Calibration of Naive Bayes' Posterior Estimates (2000). Technical Report CMU-CS-00-155, Computer Science Department, Carnegie Mellon University.

[4] R. Caruana, A. Niculescu-Mizil (2004). Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *Proceedings of the Tenth International Conference on Knowledge Discovery and Data Mining* (KDD-2004), 69-78. ACM Press.

[5] C. L. Blake and C. J. Merz (1998). UCI Repository of machine learning databases `[http://www.ics.uci.edu/~mlearn/MLRepository]`. Irvine, CA: University of California, Department of Information and Computer Science.

[6] Y. Freund and R. E. Schapire (1997). A decision-theoretic generalization of on-line learning and an application to boosting. In *Journal of Computer and System Sciences*, 55:1, 119-139.

[7] O. M. Halck (2002). Using Hard Classifiers to Estimate Conditional Class Probabilities. In *Proceedings of the European Conference on Machine Learning* (ECML-2002), 124-134, Springer.

[8] S. Hettich and S. D. Bay (1999). The UCI KDD Archive `[http://kdd.ics.uci.edu]`. Irvine, CA: University of California, Department of Information and Computer Science.

[9] Y. LeCun, L. Bottou, G. Orr, and K. Muller (1998). Efficient BackProp. In G. Orr and K. Muller (eds.) *Neural Networks: Tricks of the trade*. Springer-Verlag.

[10] D. Margineantu (2000). On class probability estimates and cost-sensitive evaluation of classifiers. In *Workshop on Cost-Sensitive Learning, The Seventeenth International Conference on Machine Learning* (ICML-2000).

[11] J. Platt (1999). Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In A. Smola, P. Bartlett, B. Schölkopf, D. Schuurmans (eds.) *Advances in Large Margin Classifiers*, 61-74. MIT Press.

[12] F. Provost and P. Domingos (2003). Tree Induction for Probability-based Rankings. In *Machine Learning*, 52:3, 199-216. Kluwer.

[13] B. Schölkopf and A. J. Smola (2002). *Learning with Kernels*. MIT Press.

[14] J. R. Quinlan (1993). *C4.5: Programs for Machine Learning*. San Mateo CA: Morgan Kaufmann.

[15] I. H. Witten and E. Frank (1999). *Data mining: practical machine learning tools and techniques with Java implementations* `[http://www.cs.waikato.ac.nz/ml/weka/]`. Morgan Kaufmann.

[16] B. Zadrozny and C. Elkan (2001). Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning* (ICML-01), 609-616. Morgan Kaufmann.

[17] B. Zadrozny, J. Langford, and N. Abe (2003). Cost Sensitive Learning by Cost-Proportionate Example Weighting. In *Proceedings of the 2003 IEEE International Conference on Data Mining* (ICDM-03), 435-442. IEEE Press.

Loss Functions for Discriminative Training of Energy-Based Models.

Yann LeCun and Fu Jie Huang
The Courant Institute, New York University
{yann,jhuangfu }@cs.nyu.edu
http://yann.lecun.com

Abstract

Probabilistic graphical models associate a probability to each configuration of the relevant variables. *Energy-based models* (EBM) associate an energy to those configurations, eliminating the need for proper normalization of probability distributions. Making a decision (an inference) with an EBM consists in comparing the energies associated with various configurations of the variable to be predicted, and choosing the one with the smallest energy. Such systems must be trained discriminatively to associate low energies to the desired configurations and higher energies to undesired configurations. A wide variety of loss function can be used for this purpose. We give sufficient conditions that a loss function should satisfy so that its minimization will cause the system to approach to desired behavior. We give many specific examples of suitable loss functions, and show an application to object recognition in images.

1 Introduction

Graphical Models are overwhelmingly treated as probabilistic generative models in which inference and learning are viewed as probabilistic estimation problems. One advantage of the probabilistic approach is *compositionality*: one can build and train component models separately before assembling them into a complete system. For example, a Bayesian classifier can be built by assembling separately-trained generative models for each class. But if a model is trained discriminatively *from end to end* to make decisions, mapping raw input to ultimate outputs, there is no need for compositionality. Some applications require hard decisions rather than estimates of conditional output distributions. One example is mobile robot navigation: once trained, the robot *must* turn left or right when facing an obstacle. Computing a distribution over steering angles would be of little use in that context. The machine should be trained from end-to-end to approach the best possible decision in the largest range of situations.

Another implicit advantage of the probabilistic approach is that it provides well-justified loss functions for learning, e.g. maximum likelihood for generative models, and max conditional likelihood for discriminative models. Because of the normalization, maximizing the likelihood of the training samples will automatically decrease the likelihood of other points, thereby driving machine to approach the desired behavior. The downside is that the negative log-likelihood is *the only well-justified loss functions*. Yet, approximating a distribution over the entire space by maximizing likelihood may be an overkill when the ultimate goal is merely to produce the right decision.

We will argue that using proper probabilistic models, because they must be normalized, considerably restricts our choice of model architecture. Some desirable architectures may be difficult to normalize (the normalization may involve the computation of intractable partition functions), or may even be non-normalizable (their partition function may be an integral that does not converge).

This paper concerns a more general class of models called *Energy-Based Models* (EBM). EBMs associate an (unnormalized) energy to each configuration of the variables to be modeled. Making an inference with an EBM consists in searching for a configuration of the variables to be predicted that minimizes the energy, or comparing the energies of a small number of configurations of those variables. EBMs have considerable advantages over traditional probabilistic models: (1) There is no need to compute partition functions that may be intractable; (2) because there is no requirement for normalizability, the repertoire of possible model architectures that can be used is considerably richer.

Training an EBM consists in finding values of the trainable parameter that associate *low energies* to "desired" configurations of variables (e.g. observed on a training set), and *high energies* to "undesired" configurations. With properly normalized probabilistic models, increasing the likelihood of a "desired" configuration of variables will automatically decrease the likelihoods of other configurations. With EBMs, this is not the case: making the energy of desired configurations low may not necessarily make the energies of other configurations high. Therefore, one must be very careful when designing loss functions for EBMs [1].

[1] it is important to note that the *energy* is quantity minimized

We must make sure that the loss function we pick will effectively drive our machine to approach the desired behavior. In particular, we must ensure that the loss function has no trivial solution (e.g. where the best way to minimize the loss is to make the energy constant for all input/output pair). A particular manifestation of this is the so-called *collapse problem* that was pointed out in some early works that attempted to combined neural nets and HMMs [7, 2, 8].

This energy-based, end-to-end approach to learning has been applied with great success to sentence-level handwriting recognition in the past [10]. But there has not been a general characterization of "good" energy functions and loss functions. The main point of this paper is to give sufficient conditions that a discriminative loss function must satisfy, so that its minimization will carve out the energy landscape in input/output space in the right way, and cause the machine to approach the desired behavior. We then propose a wide family of loss functions that satisfy these conditions, independently of the architecture of the machine being trained.

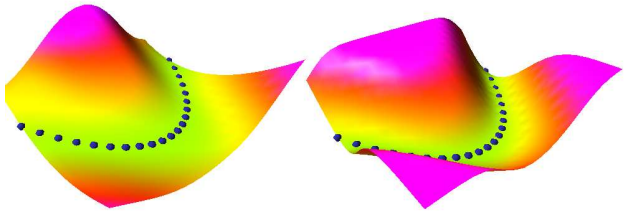


Figure 1: Two energy surfaces in X, Y space obtained by training two neural nets to compute the function $Y = X^2 - 1/2$. The blue dots represent a subset of the training samples. In the left diagram, the energy is quadratic in Y, therefore its exponential is integrable over Y. This model is equivalent to a probabilistic Gaussian model of $P(Y|X)$. The right diagram uses a non-quadratic saturated energy whose exponential is not integrable over Y. This model is not normalizable, and therefore has no probabilistic counterpart.

2 Energy-Based Models

Let us define our task as one of predicting the best configuration of a set of variables denoted collectively by Y, given a set of observed (input) variables collectively denoted by X. Given an observed configuration for X, a probabilistic model (e.g. a graphical model) will associate a (normalized) probability $P(Y|X)$ to each possible configuration of Y. When a decision must be made, the configuration of Y that maximizes $P(Y|X)$ will be picked.

An *Energy-Based Model* (EBM) associates a scalar *energy* $E(W, Y, X)$ to each configuration of X, Y. The family of possible energy functions is parameterized by a parameter vector W, which is to be learned. One can view this energy function as a measure of "compatibility" between the values of Y and X. Note that there is no requirement for normalization.

The *inference* process consists in clamping X to the observed configuration (e.g. an input image for image classification), and searching for the configuration of Y in a set $\{Y\}$ that minimizes the energy. This optimal configuration is denoted $\check{Y}$: $\check{Y} = \operatorname{argmin}_{Y \in \{Y\}} E(W, Y, X)$. In many situations, such as classification, $\{Y\}$ will be a discrete set, but in other situations $\{Y\}$ may be a continuous set (e.g. a compact set in a vector space). This paper will not discuss how to perform this inference efficiently: the reader may use her favorite and most appropriate optimization method depending upon the form of $E(W, Y, X)$, including exhaustive search, gradient-based methods, variational methods, (loopy) belief propagation, dynamic programming, etc.

Because of the absence of normalization, EBMs should only be used for *discrimination* or *decision* tasks where only the *relative energies* of the various configurations of Y for a given X matter. However, if $\exp(-E(W, Y, X))$ is integrable over Y, for all X and W, we can turn an EBM into an equivalent probabilistic model by posing:

$$P(Y|X, W) = \frac{\exp(-\ E(W, Y, X)}{\int_y \exp(-\ E(W, y, X))}$$

where is an arbitrary positive constant. The normalizing term (the denominator) is called the *partition function*. However, the EBM framework gives us more flexibility because it allows us to use energy functions whose exponential is not integrable over the domain of Y. Those models have no probabilistic equivalents.

Furthermore, we will see that training EBMs with certain loss functions circumvents the requirement for evaluating the partition function and its derivatives, which may be intractable. Solving this problem is a major issue with probabilistic models, if one judges by the considerable amount of recent publications on the subject.

Probabilistic models are generally trained with the maximum likelihood criterion (or equivalently, the negative log-likelihood loss). This criterion causes the model to approach the conditional density $P(Y|X)$ over the entire domain of Y for each X. With the EBM framework, we allow ourselves to devise loss functions that merely cause the system to make the best decisions. These loss functions are designed to place $\min_{Y \in \{Y\}} E(W, Y, X)$ near the desired Y for each X. This is a considerably less complex and less constrained problem than that of estimating the "correct" conditional density over Y for each X. To convince ourselves of this, we can note that many different energy functions may have minima at the same Y for a given X, but only one of those (or a few) maximizes the likelihood. For example, figure 1 shows two energy surfaces in X, Y space. They were obtained by training two neural nets (denoted $G(W, X)$) to approximate the function $Y = X^2 - 1/2$. In the left diagram, the energy $E(W, Y, X) = (Y - G(W, X))^2$ is quadratic in Y, there-

during inference, while the *loss* is the quantity minimized during learning

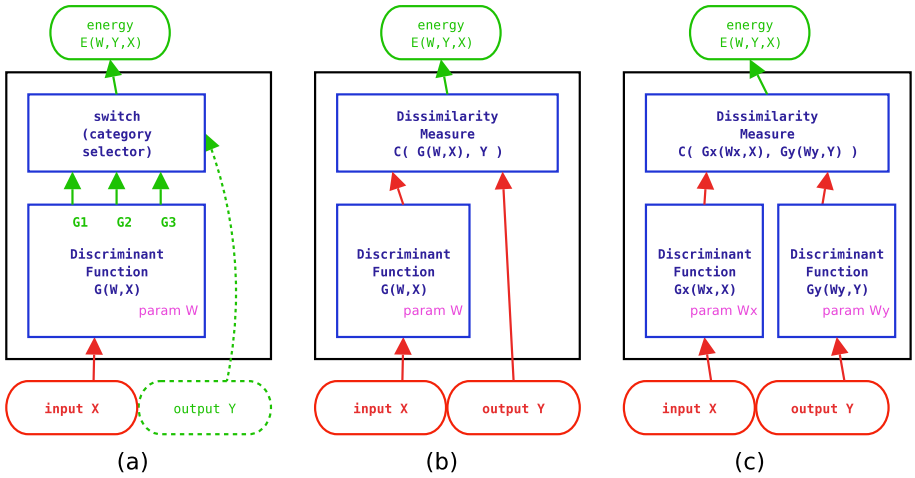


Figure 2: Examples of EBMs. (a) switch-based classifier; (b) a regressor; (c) constraint satisfaction architecture. Multidimensional variables are in red, scalars in green, and discrete variables in green dotted lines.

fore its exponential is integrable over Y. This model is equivalent to a probabilistic Gaussian model for $P(Y|X)$. The right diagram uses a non-quadratic saturated energy $E(W,Y,X) = \tanh\left((Y - G(W,X))^2\right)$ whose exponential is not integrable over Y. This model is not normalizable, and therefore has no probabilistic counterpart, yet it fulfills our desire to produce the best Y for any given X.

2.1 Previous work

Several authors have previously pointed out the shortcomings of normalized models for discriminative tasks. Bottou [4] first noted that discriminatively trained HMMs are unduly restricted in their expressive power because of the normalization requirements, a problem recently named "label bias" in [9]. To alleviate this problem, late normalization schemes for un-normalized discriminative HMMs were proposed in [6] and [10]. Recent works have revived the issue in the context of sequence labelling [5, 1, 14].

Some authors have touted the use of various non-probabilistic loss functions, such as the Perceptron loss or the maximum margin loss, for training decision-making systems [7, 10, 5, 1, 14]. However, the loss functions in these systems are intimately linked to the underlying architecture of the machine being trained. Some loss functions are incompatible with some architectures and may possess undesirable minima. The present paper gives conditions that "well-behaved" loss functions should satisfy.

Teh et al. [15] have introduced the term "Energy-Based Model" in a context similar to ours, but they only considered the log-likelihood loss function. We use the term EBM in a slightly more general sense, which include the possibility of using other loss functions. Bengio et al. [3] describe an energy-based language model, but they do not discuss the issue of loss functions.

2.2 Examples of EBMs

EBM for Classification: Traditional multi-class classifiers can be viewed as particular types of EBMs whose architecture is shown in figure 2(a). A parameterized discriminant function $G(W,X)$ produces an output vector with one component for each of the k categories $(G_0, G_1 ..., G_{k-1})$. Component G_i is interpreted as the energy (or "penalty") for assigning X to the i-th category. A discrete switch module selects which of the components is connected to the output energy. The position of the switch is controlled by the discrete variable Y, which is interpreted as the category. The output energy is equal to $E(W,Y,X) = \sum_{i=0}^{k-1} \delta(Y - i)G(W,X)_i$, where $\delta(Y-i)$ is equal to 1 for $Y = i$ and 0 otherwise (Kronecker function), and $G(W,X)_i$ is the i-th component of $G(W,X)$. Running the machine consists in finding the position of the switch (the value of Y) that minimizes the energy, i.e. the position of the switch that selects the smallest component of $G(W,X)$.

EBM for Regression: A regression function $G(W,X)$ with the squared error loss (e.g. a traditional neural network) is a trivial form of minimum energy machine (see figure 2(b)). The energy function of this machine is defined as $E(W,Y,X) = \frac{1}{2}||G(W,X) - Y||^2$. The value of Y that minimizes E is simply equal to $G(W,X)$. Therefore, running such a machine consists simply in computing $G(W,X)$ and copying the value into Y. The energy is then zero. This architecture can be used for classification by simply making $\{Y\}$ a discrete set (with one element for each category).

EBM for Constraint Satisfaction: sometimes, the dependency between X and Y cannot be expressed as a function that maps X's to Y's (consider for example the constraint $X^2 + Y^2 = 1$). In this case, one can resort to modeling "constraints" that X and Y must satisfy. The energy function measures the price for violating the constraints. An example architecture is shown in figure 2(c). The energy is $E(W,Y,X) = C(G_x(W_x,X), G_y(W_y,Y))$, where G_X

and G_y are functions to be learned, and $C(a, b)$ is a dissimilarity measure (e.g. a distance).

2.3 Deterministic Latent Variables

Many tasks are more conveniently modeled by architectures that use *latent variables*. Deterministic latent variables are extra variables (denoted by Z) that influence the energy, and that are not observed. During an inference, the energy is minimized over Y and Z:

$$(\check{Y}, \check{Z}) = \text{argmin}_{Y \in \{Y\},\, Z \in \{Z\}} E(W, Y, Z, X)$$

By simply redefining our energy function as:

$$\check{E}(W, Y, X) = \min_{Z \in \{Z\}} E(W, Y, Z, X)$$

We can essentially ignore the issue of latent variables.

Latent variables are very useful in situations where a hidden characteristic of the process being modeled can be inferred from observations, but cannot be predicted directly. This occurs for example in speech recognition, handwriting recognition, natural language processing, and biological sequence analysis, and other sequence labeling tasks where a *segmentation* must be performed simultaneously with the recognition. Alternative segmentations are often represented as paths in a weighted lattice. Each path may be associated with a category [10, 5, 1, 14]. The path being followed in the lattice can be viewed as a discrete latent variable. Searching for the best path using dynamic programming (Viterbi) or approximate methods (e.g. beam search) can be seen as a minimization of the energy function with respect to this discrete variable. For example, in a speech recognition context, evaluating $\min_{Z \in \{Z\}} E(W, Y^i, Z, X^i)$ is akin to "constrained segmentation": finding the best path in the lattice that produces a particular output label Y^i.

An EBM framework with which to perform graph manipulations and search, while preserving the ability to compute partial derivatives for learning is the Graph Transformer Network model described in [10]. However, that paper only mentions two loss functions (generalized perceptron and log-likelihood), without a general discussion of how to construct appropriate loss functions.

Rather than give a detailed description of latent-variable EBM for sequence processing, we will describe an application to visual object detection and recognition. The architecture is shown in figure 3. The input image is first turned into an appropriate representation (e.g. a feature vector) by a trainable front-end module (e.g. a convolutional network, as in [11]). This representation is then matched to models of each category. Each model outputs an energy that measures how well the representation matched the category (a low energy indicates a good match, a high energy a bad match). The switch selects the best-matching category. The object models take in latent variables that may be used to represent some instantiation parameters of the objects, such as the pose, illumination, or conformation. The optimal value of those parameters for a particular input is computed as part of the energy-minimizing inference process. Section 4 reports experimental results obtained with such a system.

3 Loss Functions for EBM Training.

In supervised learning, the *training set* $\mathcal{S}$ is a set of pairs $\mathcal{S} = \{(X^i, Y^i)\ ,\ i = 1..P\}$ where X^i is an input, and Y^i is a desired output to be predicted. Practically every learning methods can be described as the process of finding the parameter $W \in \{W\}$ that minimizes a judiciously chosen *loss function* $\mathcal{L}(W, \mathcal{S})$. The loss function should be a measure of the discrepancy between the machine's behavior and the desired behavior on the training set. Well-behaved loss functions for EBMs should shape the energy landscape so as to "dig holes" at (X, Y) locations near training samples, while "building hills" at un-desired locations, particularly the ones that are erroneously picked by the inference algorithm. For example, a good loss function for the regression problem depicted in figure 1 should dig holes around the blue dots (which represent a subset of the training set, while ensuring that the surrounding areas have higher energy.

In the following, we characterize the general form of loss functions whose minimization will make the machine carve out the energy landscape in the right way so as to approach the desired behavior. We define the loss on the full training set as:

$$\mathcal{L}(W, \mathcal{S}) = R\left(\frac{1}{P}\sum_{i=1}^{P} L(W, Y^i, X^i)\right) \qquad (1)$$

where $L(W, Y^i, X^i)$ is the per-sample loss function for sample (X^i, Y^i), and R is a monotonically increasing function. Loss functions that combine per-sample losses through other symmetric n-ary operations than addition (e.g. multiplication, as in the case of likelihood-like loss functions) can be trivially obtained from the above through judicious choices of R and L. With this definition, the loss is invariant under permutations of the samples, and under multiple repetitions of the same training set. In the following we will set R to the identity function. We assume that $L(W, Y^i, X^i)$ has a lower bound over W for all Y^i, X^i.

At this point, we can note that if we minimize *any* such loss on a training set over a set of functions with finite VC-dimension, appropriate VC-type upper bounds for the expected loss will apply, ensuring the convergence of the empirical loss to the expected loss as the training set size increases. Therefore, we will only discuss the conditions under which a loss function will make the machine approach the desired behavior *on the training set*.

Sometimes, the task uniquely defines a "natural" loss function (e.g. the number of mis-classified examples), but more often than not, minimizing that function is impractical. Therefore one must resort to surrogate loss functions whose choice is up to the designer of the system. One crucial, but often neglected, question must be answered before choosing a loss function: "will minimizing the loss cause the

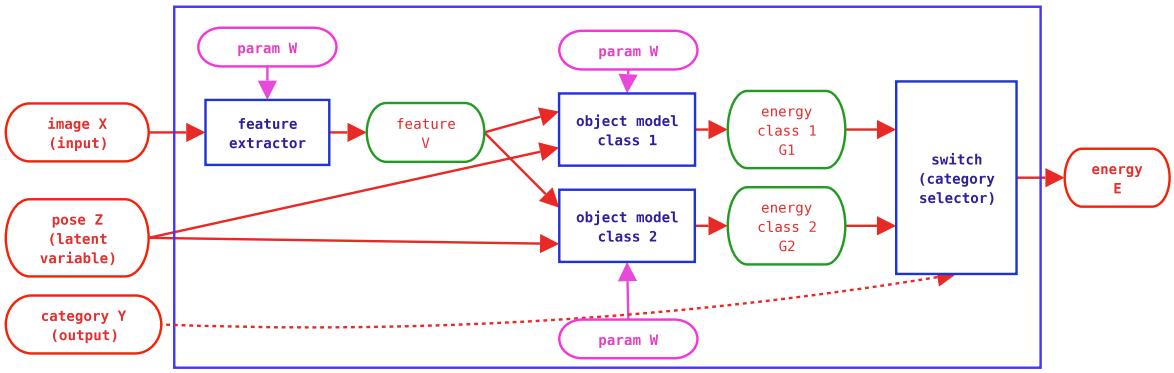


Figure 3: Example of a switch-based Energy-Based Model architecture for object recognition in images, where the pose of the object is treated as a latent variable.

learning machine to approach the desired behavior?" We will give general conditions for that.

The inference process produces the Y that minimizes $\check{E}(W, Y, X^i)$. Therefore, a well-designed loss function must drive the energy of the desired output $\check{E}(W, Y^i, X^i)$ to be lower than the energies of all the other possible outputs. Minimizing the loss function should result in "holes" at X, Y locations near the training samples, and "hills" at un-desired locations.

3.1 Conditions on the Energy

The condition for the correct classification of sample X^i is:

Condition 1 $\check{E}(W, Y^i, X^i) < \check{E}(W, Y, X^i)$, $\forall Y \in \{Y\}, Y \neq Y^i$

To ensure that the correct answer is robustly stable, we may choose to impose that the energy of the desired output be lower than the energies of the undesired outputs by a *margin* m:

Condition 2 $\check{E}(W, Y^i, X^i) < \check{E}(W, Y, X^i) - m$, $\forall Y \in \{Y\}, Y \neq Y^i$

We will now consider the case where Y is a discrete variable. Let us denote by $\bar{Y}$ the output that produces the smallest energy *while being different* from the desired output Y^i: $\bar{Y} = \text{argmin}_{Y \in \{Y\}, Y \neq Y^i} \check{E}(W, Y, X^i)$. Condition 2 can be rewritten as:

Condition 3 $\check{E}(W, Y^i, X^i) < \check{E}(W, \bar{Y}, X^i) - m$, $\bar{Y} = \text{argmin}_{Y \in \{Y\}, Y \neq Y^i} \check{E}(W, Y, X^i)$

For the continuous Y case, we can simply define $\bar{Y}$ as the lowest-energy output outside of a ball of a given radius around the desired output: $\text{argmin}_{Y \in \{Y\}, ||Y - Y^i|| > \epsilon} \check{E}(W, Y, X^i)$

3.2 Sufficient conditions on the loss function

We will now make the key assumption that L depends on X^i only indirectly through the set of energies $\{\check{E}(W, Y, X^i) \ , \ Y \in \{Y\}\}$. For example, if $\{Y\}$is the set of integers between 0 and $k-1$, as would be the case for the switch-based classifier with k categories shown in figure 2(a), the per-sample loss for sample (X^i, Y^i) should be of the form:

$$L(W, Y^i, X^i) = L(Y^i, \check{E}(W, 0, X^i), \ldots, \check{E}(W, k-1, X^i)) \quad (2)$$

With this assumption, we separate the choice of the loss function from the details of the internal structure of the machine, and limit the discussion to how minimizing the loss function affects the energies.

We must now characterize the form that L can take such that its minimization will eventually drive the machine to satisfy condition 3.

We must design L in such a way that minimizing it will decrease the difference $\check{E}(W, Y^i, X^i) - \check{E}(W, \bar{Y}, X^i)$, whenever $\check{E}(W, \bar{Y}, X^i) < \check{E}(W, Y^i, X^i) + m$. In other words, whenever the difference between the energy of the incorrect answer with the lowest energy and the energy of the desired answer is less than the margin, our learning procedure should make that difference larger. We will now propose a set of sufficient conditions on the loss that guarantee this.

Since we are only concerned with how the loss influences the relative values of $\check{E}(W, Y^i, X^i)$, and $\check{E}(W, \bar{Y}, X^i)$, we will consider the shape of loss surface in the space of $\check{E}(W, Y^i, X^i)$ and $\check{E}(W, \bar{Y}, X^i)$, and view the other arguments of the loss (the energies for all the other values of Y) as parameters of that surface:

$$L(W, Y^i, X^i) = Q_{[E_y]}(\check{E}(W, Y^i, X^i), \check{E}(W, \bar{Y}, X^i))$$

where the parameter $[E_y]$ contains the vector of energies for all values of Y except Y^i and $\bar{Y}$.

We can now state sufficient conditions that guarantee that minimizing L will eventually satisfy condition 3. In all the sufficient conditions stated below, we assume that there exist a W such that condition 3 is satisfied for a single training example (X^i, Y^i), and that $Q_{[E_y]}(\check{E}(W, Y^i, X^i), \check{E}(W, \bar{Y}, X^i))$ is convex (convex in

its 2 arguments, but not necessarily convex in W). The conditions must old for all values of $[E_y]$.

Condition 4 *The minima of* $Q_{[E_y]}(\check{E}(W, Y^i, X^i), \check{E}(W, \bar{Y}, X^i))$ *are in the half-plane* $\check{E}(W, \bar{Y}, X^i) < \check{E}(W, Y^i, X^i) + m$.

This condition on the loss function clearly ensures that minimizing it will drive the machine to find a solution that satisfies condition 3, if such a solution exists.

Another sufficient condition can be stated to characterize loss functions that do not have minima, or whose minimum is at infinity:

Condition 5 *the gradient of* $Q_{[E_y]}(\check{E}(W, Y^i, X^i), \check{E}(W, \bar{Y}, X^i))$ *on the margin line* $\check{E}(W, \bar{Y}, X^i) = \check{E}(W, Y^i, X^i) + m$, *has a positive dot product with the direction [-1,1].*

This condition guarantees that minimizing L will drive the energies $\check{E}(W, \bar{Y}, X^i)$ and $\check{E}(W, Y^i, X^i)$ toward the half-plane $\check{E}(W, \bar{Y}, X^i) < \check{E}(W, Y^i, X^i) + m$.

Yer another sufficient condition can be stated to characterize loss functions whose minima are not in the desired half-plane, but where the possible values of $\check{E}(W, \bar{Y}, X^i)$ and $\check{E}(W, Y^i, X^i)$ are constrained by their dependency on W in such a way that the minimum of the loss *while satisfying the constraint* is in the desired half-plane:

Condition 6 *On the margin line* $\check{E}(W, \bar{Y}, X^i) = \check{E}(W, Y^i, X^i) + m$, *the following must hold:* $\left[\frac{\partial \check{E}(W,Y^i,X^i)}{\partial W} - \frac{\partial \check{E}(W,\bar{Y},X^i)}{\partial W}\right] . \frac{\partial L(W,Y^i,X^i)}{\partial W} > 0$

This condition ensures that an update of the parameters W to minimize the loss will drive the energies $\check{E}(W, \bar{Y}, X^i)$ and $\check{E}(W, Y^i, X^i)$ toward the desired half-plane $\check{E}(W, \bar{Y}, X^i) < \check{E}(W, Y^i, X^i) + m$.

We must emphasize that these are only sufficient conditions. There may be legitimate loss functions (e.g. non-convex functions) that do not satisfy them, yet have the proper behavior.

3.3 Examples of Loss Functions

We can now give examples of loss functions that satisfy the above criteria, and examine whether some of the popular loss functions proposed in the literature satisfy it.

Energy Loss: The simplest and most widely used loss is the *energy loss*, which is simply of the form $L_{\text{energy}}(W, Y^i, X^i) = \check{E}(W, Y^i, X^i)$. This loss does not satisfy condition 4 or 5 in general, but there are certain forms of $\check{E}(W, Y^i, X^i)$ for which condition 6 is satisfied. For example, let us consider an energy of the form $E(W, Y, X) = \sum_{k=1}^{K} \delta(Y - k).||U^k - G(W, X)||^2$. Function $G(W, X)$ could be a neural net, on top of which are placed K radial basis functions whose centers are the vectors U^k. If the U^k are fixed and all different, then the energy loss applied to this machine fulfills condition 6. Intuitively, that is because by pulling $G(W, X)$ towards one of the RBF centers, we push it away from the others. Therefore when the energy of the desired output decreases, the other ones increase. However, if we allow the RBF centers to be learned, condition 6 is no longer fulfilled. In that case, the loss has spurious minima where all the RBF centers are equal, and the function $G(W, X)$ is constant and equal to that RBF center. The loss is zero, but the machine does not produce the desired result. Picking any combination of loss and energy that satisfy any of the conditions 4, 5, or 6 solves this collapse problem.

Generalized Perceptron Loss: We define the generalized Perceptron loss for training sample (X^i, Y^i) as:

$$L_{\text{ptron}}(W, Y^i, X^i) = \check{E}(W, Y^i, X^i) - \min_{Y \in \{Y\}} \check{E}(W, Y, X^i) \tag{3}$$

It is easy to see that with $E(W, Y^i, X^i) = -Y^i.W^T X^i$, and $\{Y\} = \{-1, 1\}$, the above loss reduces to the traditional linear Perceptron loss. The generalized perceptron loss satisfies condition 4 with $m = 0$. This loss was used by [10] for training a commercially deployed handwriting recognizer that combined a heuristic segmenter, a convolutional net, and a language model (where the latent variables represented paths in an interpretation lattice). A similar loss was studied by [5] for training a text parser. Because the margin is zero, this loss may not prevent collapses for certain architecture.

Generalized Margin Loss: A more robust version of the Perceptron loss is the *Margin Loss*, which directly uses the energy of most offending non-desired output $\bar{Y}$ in the contrastive term:

$$L_{\text{margin}}(W, Y^i, X^i) = Q_m[\check{E}(W, Y^i, X^i) - \check{E}(W, \bar{Y}, X^i)] \tag{4}$$

where $Q_m(e)$ is any function that is monotonically increasing for $e > -m$. The traditional "hinge loss" used with kernel-based methods, and the loss used by the LVQ2 algorithm are special cases with $Q_m(e) = e + m$ for $e > -m$, and 0 otherwise. Special forms of that loss were used in [7] for discriminative speech recognition, and in [1] and [14] for text labeling. The exponential loss used in AdaBoost is a special form of equation (4) with $Q_m(e) = \exp(e)$. A slightly more general form of the margin loss that satisfies conditions 4 or 5 is given by:

$$L_{\text{gmargin}}(W, Y^i, X^i) = Q_{gm}[\check{E}(W, Y^i, X^i), \check{E}(W, \bar{Y}, X^i)] \tag{5}$$

with the condition that $\frac{\partial Q_{gm}(e1,e2)}{\partial e1} > \frac{\partial Q_{gm}(e1,e2)}{\partial e2}$ when $e1 + m > e2$. An example of such loss is:

$$L(W, Y^i, X^i) = Q^+(\check{E}(W, Y^i, X^i)) + Q^-(\check{E}(W, \bar{Y}, X^i)) \tag{6}$$

where $Q^+(e)$ is a convex monotonically increasing function, and $Q^-(e)$ a convex monotonically decreasing function such that if $\frac{dQ^+}{de}|e1 = 0$ and $\frac{dQ^-}{de}|e2 = 0$ then $e1 + m < e2$. When $\check{E}(W, Y^i, X^i)$ is akin to a distance (bounded below by 0), a judicious choice for Q^+ and Q^-

is:

$$L(W, Y^i, X^i) = \check{E}(W, Y^i, X^i)^2 + \kappa \exp(- \check{E}(W, \bar{Y}, X^i)) \quad (7)$$

where κ and are positive constants. A similar loss function was recently used by our group to train a pose-invariant face detection [12]. This system can simultaneously detect faces and estimate their pose using latent variables to represent the head pose.

Contrastive Free Energy Loss: While the loss functions proposed thus far involve only $\check{E}(W, \bar{Y}, X^i)$ in their contrastive part, loss functions can be devised to combine all the energies for all values of Y in their contrastive term:

$$L(W, Y^i, X^i) = \check{E}(W, Y^i, X^i) - F(\check{E}(W, 0, X^i), \ldots, \check{E}(W, k-1, X^i)) \quad (8)$$

F can be interpreted as a *generalized free energy* of the ensemble of systems with energies $\check{E}(W, Y, X^i)\ \forall Y \in \{Y\}$. It appears difficult to characterize the general form of F that ensures that L satisfies condition 5. An interesting special case of this loss is the familiar **negative log-likelihood loss**:

$$L_{\text{nll}}(W, Y^i, X^i) = \check{E}(W, Y^i, X^i) - F_\beta(W, X^i) \quad (9)$$

with

$$F_\beta(W, X^i) = -\frac{1}{} \log \left(\int_{Y \in \{Y\}} \exp[- \check{E}(W, Y, X^i)] \right) \quad (10)$$

where is a positive constant. The second term can be interpreted as the Helmholtz free energy (log partition function) of the ensemble of systems with energies $\check{E}(W, Y, X^i)\ \forall Y \in \{Y\}$. This type of discriminative loss with $= 1$ is widely used for discriminative probabilistic models in the speech, handwriting, and NLP communities [10, 2, 8]. This is also the loss function used in the *conditional random field* model of Lafferty et al [9].

We can see that loss (9) reduces to the generalized Perceptron loss when $\to \infty$. Computing this loss and its derivative requires computing integrals (or sums) over $\{Y\}$ that may be intractable. It also assumes that the exponential of the energy be integrable over $\{Y\}$, which puts restrictions on the choice of $E(W, Y, X)$ and/or $\{Y\}$.

4 Illustrative Experiments

To illustrate the use of contrastive loss functions with non-probabilistic latent variables, we trained a system to recognize generic objects in images independently of the pose and the illumination. We used the NORB dataset [11] which contains 50 different uniform-colored toy objects under 18 azimuths, 9 elevations, and 6 lighting conditions. The objects are 10 instance from 5 generic categories: four-legged animals, human figures, airplanes, trucks, and cars. Five instances of each category were used for training, and the other five for testing (see figure 4). A 6-layer convolutional network trained with the mean-square loss achieves

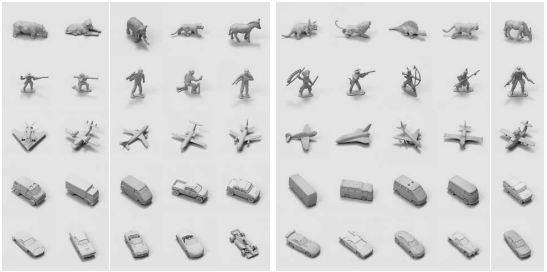


Figure 4: Invariant object recognition with NORB dataset. The left portion shows sample views of the training instances, and the right portion testing instances for the 5 categories.

6.8% test error on this set when fed with binocular 96×96-pixel gray-scale images [11].

We used an architecture very much like the one in figure 3, where the feature extraction module is identical to the first 5 layers of the 6-layer convolutional net used in the reference experiment. The object model functions were of the form: $E_i = ||W_i.V - F(Z)||$, $i = 1..5$, where V is the output of the 5-layer net (100 dimensions), W_i is a 9×100 (trainable) weight matrix. The latent variable Z has two dimensions that are meant to represent the azimuth and elevation of the object viewpoint. The set of possible values $\{Z\}$ contained 162 values (azimuths: 0-360 degrees every 20, elevations: 30-70 degrees every 5). The output of $F(Z)$ is a point on an azimuth/elevation half-sphere (2D surface) embedded in the 9D hypercube $[-1, +1]^9$. The minimization of the energy over Z is performed through exhaustive search (which is relatively cheap).

We used the loss function (7). This loss causes the convolutional net to produce a point as close as possible to any point on the half-sphere of the desired class, and as far as possible from the half-sphere of the best-scoring non-desired class. The system is trained "from scratch" including the convolutional net. We obtained 6.3% error on the test set, which is a moderate, but significant improvement over the 6.8% of the control experiment.

5 Discussion

Efficient End-to-End Gradient-Based Learning To perform gradient-based training of all the modules in the architecture, we must compute the gradient of the loss with respect to all the parameters. This is easily achieved with the module-based generalization of back-propagation described in [10]. A typical learning iteration would involve the following steps: (1) one forward propagation through the modules that only depend on X; (2) a run of the energy-minimizing inference algorithm on the modules that depend on Z and Y; (3) as many back-propagations through the modules that depend on Y as there are energy terms in the loss function; (4) one back-propagation through the module that depends only on X; (5) one update of the parameters.

Efficient Inference: Most loss functions described in this paper involve multiple runs of the machine (in the worst case, one run for each energy term that enters in the loss). However, the parts of the machine that solely depend on X, and not on Y or Z need not be recomputed for each run (e.g. the feature extractor in figure 3), because X does not change between runs.

If the energy function can be decomposed as a sum of functions (called *factors*) $E(W, Y, X) = \sum_j E_j(W_j, Y, Z, X)$, each of which takes subsets of the variables in Z and Y as input, we can use a form of belief propagation algorithm for factor graphs to compute the lowest energy configuration [13]. These algorithms are exact and tractable if Z and Y are discrete and the factor graph has no loop. They reduce to Viterbi-type algorithms when members of $\{Z\}$ can be represented by paths in a lattice (as is the case for sequence segmentation/labeling tasks).

Approximate Inference: We do not really need to assume that the energy-minimizing inference process always finds the global minimum of $E(W, Y, X^i)$ with respect to Y. We merely need to assume that this process finds approximate solutions (e.g. local minima) in a consistent, repeatable manner. Indeed, if there are minima of $E(W, Y, X^i)$ with respect to Y that our minimization/inference algorithm never finds, we do not need to find them and increase their energy. Their existence is irrelevant to our problem. However, it is important to note that those unreachable regions of low energy will affect the loss function of probabilistic models as well as that of EBMs if the negative log-likelihood loss is used. This is a distinct advantage of the un-normalized EBM approach: low-energy areas that are never reached by the inference algorithm are not a concern.

6 Conclusion and Outlook

Most approaches to discriminative training of graphical models in the literature use loss functions from a very small set. We show that energy-based (un-normalized) graphical models can be trained discriminatively using a very wide family of loss functions. We give a sufficient condition that the loss function must satisfy so that its minimization will make the system approach the desired behavior. We give a number of loss functions that satisfy this criterion and describe experiments in image recognition that illustrate the use of such discriminative loss functions in the presence of non-probabilistic latent variables.

Acknowledgments

The authors wish to thank Leon Bottou, Yoshua Bengio, Margarita Osadchy, and Matt Miller for useful discussions.

References

[1] Yasemin Altun, Mark Johnson, and Thomas Hofmann. Loss functions and optimization methods for discriminative learning of label sequences. In *Proc. EMNLP*, 2003.

[2] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe. Global optimization of a neural network-hidden Markov model hybrid. *IEEE Transaction on Neural Networks*, 3(2):252–259, 1992.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, February 2003.

[4] L. Bottou. *Une Approche théorique de l'Apprentissage Connexionniste: Applications à la Reconnaissance de la Parole*. PhD thesis, Université de Paris XI, 91405 Orsay cedex, France, 1991.

[5] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*, 2002.

[6] J. S. Denker and C. J. Burges. Image segmentation and recognition. In *The Mathematics of Induction*. Addison Wesley, 1995.

[7] X. Driancourt and L. Bottou. MLP, LVQ and DP: Comparison & cooperation. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 815–819, Seattle, 1991.

[8] P. Haffner. Connectionist speech recognition with a global MMI algorithm. In *Eurospeech'93*, Berlin, September 1993.

[9] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. International Conference on Machine Learning (ICML)*, 2001.

[10] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[11] Yann LeCun, Fu-Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proc. CVPR*, 2004.

[12] M. Osadchy, M. Miller, and Y. LeCun. Synergistic face detection and pose estimation. In *Proc. NIPS*, 2004.

[13] Kschischang F. R., B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Information Theory*, 47(2):498–519, February 2001.

[14] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *Proc. NIPS*, 2003.

[15] Y. W. Teh, M. Welling, S. Osindero, and Hinton G. E. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4:1235–1260, 2003.

Probabilistic Soft Interventions in Conditional Gaussian Networks

Florian Markowetz, Steffen Grossmann, and Rainer Spang
firstname.lastname@molgen.mpg.de
Dept. Computational Molecular Biology
Max Planck Institute for Molecular Genetics
Berlin, Germany

Abstract

We introduce a general concept of probabilistic interventions in Bayesian networks. This generalizes deterministic interventions, which fix nodes to certain states. We propose "pushing" variables in the direction of target states without fixing them. We formalize this idea in a Bayesian framework based on Conditional Gaussian networks.

1 Introduction

In modern biology, the key to infering gene function and regulatory pathways are experiments with interventions into the normal course of action in a cell. A common technique is to perturb a gene of interest experimentally and to study which other genes' activity or phenotypic features are effected. Bayesian networks present a prominent approach to derive a theoretical model from these experiments (Pe'er et al., 2001; Yoo et al., 2002; Friedman, 2004): genes are represented by vertices of a network and the task is to find a topology, which explains dependencies between the genes. When learning from observational data only, groups of Bayesian networks may be statistically indistinguishable (Verma and Pearl, 1990). Information about effects of an intervention helps to resolve such equivalence classes by including causal knowledge into the model (Tian and Pearl, 2001). The final goal is to learn a graph structure which not only represents statistical dependencies, but also causal relations between genes.

Manipulating the expression level of a gene can be done in a variety of ways (Alberts et al., 2002). A gene's expression level can be down-regulated by several techniques including

1. creating animals or cell lines in which the gene is non-functional. This is called a *knockout*.
2. exposing a cell or animal to environmental stress to inhibit the function of certain genes or proteins.
3. partially destroying the RNA transcribed from the gene which itself is left intact. This is the recently introduced method of *RNA interference* (RNAi).

All three examples have in common that the gene's expression level is *pushed* towards a "no expression" state. Only in the first example, however, the intervention leads to a completely unfunctional gene. In RNAi the gene is still active, but silenced. It is less active than normal due to human intervention. Hence, we do not fix the state of the gene, but push it towards lower activities. In addition this pushing is randomized to some extent: the experimentalist knows that he has silenced the gene, but he can not say exactly by how much.

It is crucial that models reflect the way data was generated in the perturbation experiments. In Bayesian structure learning, Tian and Pearl (2001) show that interventions can be modeled by imposing different parameter priors when the gene is actively perturbed or passively observed. They only distinguish between two kinds of interventions: most generally, interventions that change the local probability distribution of the node within a given family of distributions, and as a special case, interventions that fix the state of the variable deterministically. The first is called a *mechanism change*; it does not assume any prior information on *how* the local probability distribution changes. The second type of intervention, which fixes the state of the variable, is called a *do-operation* and is treated in detail in (Pearl, 2000; Spirtes et al., 2000). A do-operation is used in almost all applications of interventional learning in Bayesian networks (e.g. Yoo and Cooper, 2003; Yoo et al., 2002; Steck and Jaakkola, 2002; Tong and Koller, 2001; Pe'er at al., 2001; Murphy, 2001; Cooper, 2000; Cooper and Yoo, 1999).

To model biological experiments as described above we focus on interventions, which specifically concen-

trate the local distribution at a certain node around some target state. We will call them *pushing interventions*, they are examples of mechanism changes with prior knowledge. The do-operator is a special case of a pushing intervention, which we call a *hard intervention*. In this paper, we generalize hard interventions to *soft interventions*: The local probability distribution only centers more around the target value without being fixed. This generalization is necessary to cope with experiments as in the gene perturbation examples 2 and 3 above. If we treat them as unfocussed mechanism changes we lose valuable information about what kind of intervention was performed. Thus, we need a concept of interventions, which is more directed than general mechanism changes, but still softer than deterministic fixing of variables.

The goal of the paper is to develop a theory for learning a Bayesian network when data from different (hard or soft) pushing interventions of the network is available. We first explain how soft interventions can be modeled by changing the prior distribution in Section 2. A soft intervention can be realized by introducing a "pushing parameter", which captures the pushing strength. We propose a concrete parametrization of the pushing parameter in the classical cases of discrete and Gaussian networks. Hard interventions, which have been formally described by choosing a Dirac prior in (Tian and Pearl, 2001), can then be interpreted as infinite pushing.

Section 3 summarizes the results in the general setting of Conditional Gaussian networks. This extends the existing theory on learning with hard interventions in discrete networks to learning with soft interventions in networks containing discrete and Gaussian variables.

The concluding Section 4 deals with *probabilistic* soft interventions: in this set-up the pushing parameter becomes a random variable and we assign a hyperprior to it. Hence, we account for the experimentalists lack of knowledge on the actual strength of intervention by weighted averaging over all possible values.

2 Pushing interventions in Bayesian networks

A Bayesian network is a graphical representation of the dependency structure between the components of a random vector $\mathbf{X}$. The individual random variables are associated with the vertices of a directed acyclic graph (DAG) D, which describes the dependency structure. Once the states of its parents are given, the probability distribution of a given node is fixed. Thus, the Bayesian network is completely specified by the DAG and the local probability distributions (LPDs).

Although this definition is quite general, there are basically three types of Bayesian networks which are used in practice: discrete, Gaussian and Conditional Gaussian (CG) networks. CG networks are a combination of the former two and will be treated in more detail in Section 3, for the rest of this section we focus on discrete and Gaussian networks.

In discrete and Gaussian networks, LPDs are taken from the family of the multinomial and normal distribution, respectively. In the theory of Bayesian structure learning, the parameters of these distributions are not fixed, but instead a prior distribution is assumed (Cooper and Herskovits, 1992; Geiger and Heckerman, 1994; Bøttcher, 2004). The priors usually chosen because of conjugacy are the Dirichlet distribution in the discrete case and the Normal-inverse-χ^2 distribution in the Gaussian case. Averaging the likelihood over these priors yields the marginal likelihood – the key quantity in structure learning (see Section 3).

An intervention at a certain node in the network can in this setting easily be modeled by a change in the LPDs' prior. When focusing on (soft) pushing interventions, this change should result in an increased concentration of the node's LPD around the target value. We model this concentration by introducing a pushing parameter w which is meant to measure the strength of the pushing — a higher value of w results in a stronger concentration of the LPD. We now explain in more detail how this is done for discrete and Gaussian networks. Since the joint distribution $p(\mathbf{x})$ in a Bayesian network factors according to the DAG structure in terms only involving a single node and its parents, it will suffice to concentrate on one such family of nodes.

2.1 Pushing by Dirichlet priors

We denote the set of discrete nodes by Δ and a discrete random variable at node $\delta \in \Delta$ by I_δ. The set of possible states of I_δ is $\mathcal{I}_\delta$. The parametrization of the discrete LPD at node δ is called θ_δ. For every configuration $\mathbf{i}_{pa(\delta)}$ of parents, θ_δ contains a vector of probabilities for each state $i_\delta \in \mathcal{I}_\delta$. Realizations of discrete random variables are multinomially distributed with parameters depending on the state of discrete parents. The conjugate prior is Dirichlet with parameters also depending on the state of discrete parents:

$$\begin{aligned} I_\delta \mid \mathbf{i}_{pa(\delta)}, \theta_\delta &\sim \text{Multin}(1, \theta_{\delta|\mathbf{i}_{pa(\delta)}}), \\ \theta_{\delta|\mathbf{i}_{pa(\delta)}} &\sim \text{Dirichlet}(\alpha_{\delta|\mathbf{i}_{pa(\delta)}}). \end{aligned} \tag{1}$$

We assume that the $\alpha_{\delta|\mathbf{i}_{pa(\delta)}}$ are chosen to respect likelihood equivalence as in (Heckerman et al., 1995). Doing a pushing intervention at node δ amounts to changing the prior parameters such that the multinomial density concentrates at some target value j. We for-

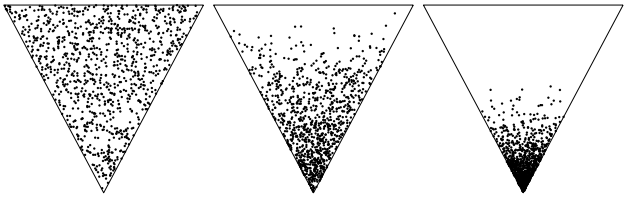


Figure 1: Examples of pushing a discrete variable with three states. Each triangle represents the sample space of the three-dimensional Dirichlet distribution (which is the parameter space of the multinomial likelihood of the node). The left plot shows a uniform distribution with Dirichlet parameter $\alpha = (1, 1, 1)$. The other two plots show effects of pushing with increasing weight: $w = 3$ in the middle and $w = 10$ at the right. In each plot 1000 points were sampled.

malize this by introducing a pushing operator $\mathcal{P}$ defined by

$$\mathcal{P}(\alpha_{\delta|\mathbf{i}_{pa(\delta)}}, w_\delta, j) \;=\; \alpha_{\delta|\mathbf{i}_{pa(\delta)}} + w_\delta \cdot \mathbf{1}_j, \tag{2}$$

where $\mathbf{1}_j$ is a vector of length $|\mathcal{I}_\delta|$ with all entries zero except for a single 1 at state j. The pushing parameter $w_\delta \in [0, \infty]$ determines the strength of intervention at node δ: if $w_\delta = 0$ the prior remains unchanged, if $w_\delta = \infty$ the Dirichlet prior degenerates to a Dirac distribution and fixes the LPD to the target state j. Figure 1 shows a three-dimensional example of increasing pushing strength w_δ.

2.2 Pushing by Normal-inverse-χ^2 priors

The set of Gaussian nodes will be called Γ and we denote a Gaussian random variable at node $\gamma \in \Gamma$ by Y_γ. In the purely Gaussian case it depends on the values of parents $\mathbf{Y}_{pa(\gamma)}$ via a vector of regression coefficients β_γ. If we assume that β_γ contains a first entry $\beta_\gamma^{(0)}$, the parent-independent contribution of Y_γ, and attach to $\mathbf{Y}_{pa(\gamma)}$ a leading 1, we can write for Y_γ the standard regression model (Bøttcher, 2004):

$$\begin{aligned} Y_\gamma \mid \beta_\gamma, \sigma_\gamma^2 \;&\sim\; \mathrm{N}(\mathbf{Y}_{pa(\gamma)}^\top \beta_\gamma,\; \sigma_\gamma^2), \\ \beta_\gamma \mid \sigma_\gamma^2 \;&\sim\; \mathrm{N}(\mathbf{m},\; \sigma_\gamma^2 \mathbf{M}^{-1}), \\ \sigma_\gamma^2 \;&\sim\; \text{Inv-}\chi^2(\nu,\; s^2). \end{aligned} \tag{3}$$

We assume that the prior parameters $\mathbf{m}, \mathbf{M}, \nu, s^2$ are chosen as in (Bøttcher, 2004). To push Y_γ to a value k we exchange $\mathbf{m}$ and s^2 by $(\mathbf{m}', s'^2) = \mathcal{P}((\mathbf{m}, s^2), w_\gamma, k)$ defined by

$$\begin{aligned} \mathbf{m}' &= e^{-w_\gamma} \cdot \mathbf{m} + (1 - e^{-w_\gamma}) \cdot k\mathbf{1}_1, \\ s'^2 &= s^2/(w_\gamma + 1), \end{aligned} \tag{4}$$

where $k\mathbf{1}_1$ is a vector of length $|\mathbf{i}_{pa(\gamma)}| + 1$ with all entries zero except the first, which is k. We use $\mathcal{P}$ for the pushing operator as in the case of discrete nodes; which one to use will be clear from the context. Again $w_\gamma \in [0, \infty]$ represents intervention strength.

The exponential function maps the real valued w into the interval $[0, 1]$. The exponential decay towards 0 ensures that by increasing w interventions quickly gain in strength. The interventional prior mean $\mathbf{m}'$ is a convex combination of the original mean $\mathbf{m}$ with a "pushing" represented by $k\mathbf{1}_1$. If $w = 0$ the mean of the normal prior and the scale of the inverse-χ^2 prior remain unchanged. As $w \to \infty$ the scale s'^2 goes to 0, so the prior for σ^2 tightens at 0. At the same time, the regression coefficients of the parents converge to 0 and β_0 goes to value k. All in all, with increasing w the distribution of Y_γ peaks more and more sharply at $Y_\gamma = k$.

Note that the discrete pushing parameter w_δ and the Gaussian pushing parameter w_γ live on different scales and will need to be calibrated individually.

2.3 Hard pushing

Hard pushing means to make sure that a certain node's LPD produces almost surely a certain target value. It has been proposed by Tian and Pearl (2001) to model this by imposing a Dirac prior on the LPD of the node. Although the Dirac prior is no direct member of neither the Dirichlet nor the Normal-inverse-χ^2 family of distributions it arises for both of them when taking the limit $w \to \infty$ for the pushing strength. Tian and Pearl (2001) give an example for discrete networks, which can easily be extended to Gaussian networks by

$$\begin{aligned} p(\beta_\gamma, \sigma_\gamma^2 \mid \mathrm{do}(Y_\gamma = k)) = \\ d(\beta_\gamma^{(0)} - k) \prod_{i \in pa(\gamma)} d(\beta_\gamma^{(i)}) \cdot d(\sigma_\gamma^2). \end{aligned} \tag{5}$$

Here, $d(\cdot)$ is the Dirac function. Averaging over this prior sets the variance and the regression coefficients to zero, while $\beta_\gamma^{(0)}$ is set to k. Thus, the marginal distribution of Y_γ is fixed to state k with probability one.

2.4 Modeling interventions by policy variables

Hard interventions can be modeled by introducing a policy variable as an additional parent node of the variable at which the intervention is occuring (Pearl, 2000; Spirtes et al., 2000; Lauritzen, 2000). In the same way we can use policy variables to incorporate soft interventions. For each node v, we introduce an additional parent node F_v ("F" for "force"), which is keeping track of whether an intervention was performed at X_v or not, and if yes, what the target state was. For a

discrete variable I_δ, the policy variable F_δ has state space $\mathcal{I}_\delta \cup \emptyset$ and we can write

$$p(\theta_{\delta|\mathbf{i}_{pa(\delta)},f_\delta}) = \\ = \begin{cases} \text{Dirichlet}(\alpha_{\delta|\mathbf{i}_{pa(\delta)}}) & \text{if } F_\delta = \emptyset, \\ \text{Dirichlet}(\alpha'_{\delta|\mathbf{i}_{pa(\delta)}}) & \text{if } F_\delta = j, \end{cases} \quad (6)$$

where $\alpha'_{\delta|\mathbf{i}_{pa(\delta)}} = \mathcal{P}(\alpha_{\delta|\mathbf{i}_{pa(\delta)}}, w_\delta, j)$ is derived from $\alpha_{\delta|\mathbf{i}_{pa(\delta)}}$ as defined in Eq. 2. For a continuous variable Y_γ, the policy variable F_γ has state space $\mathbb{R} \cup \emptyset$ and we can write

$$p(\beta_{\gamma|f_\gamma}, \sigma^2_{\gamma|f_\gamma}) = \\ = \begin{cases} \text{N}(\mathbf{m}, \mathbf{M}) \cdot \text{Inv-}\chi^2(\nu, s^2) & \text{if } F_\gamma = \emptyset, \\ \text{N}(\mathbf{m}', \mathbf{M}) \cdot \text{Inv-}\chi^2(\nu, s'^2) & \text{if } F_\gamma = k, \end{cases} \quad (7)$$

where $(\mathbf{m}', s'^2) = \mathcal{P}((\mathbf{m}, s^2), w_\gamma, k)$ as defined in Eq. 4. Equations 6 and 7 will be used in section 3.2 to compute the marginal likelihood of Conditional Gaussian networks from a mix of interventional and non-interventional data.

3 Pushing in Conditional Gaussian networks

In this section we summarize the results in the general framework of Conditional Gaussian networks and compute a scoring metric for learning from soft interventions.

3.1 Conditional Gaussian networks

Conditional Gaussian (CG) networks are Bayesian networks encoding a joint distribution over discrete and continuous variables. We consider a random vector $\mathbf{X}$ splitting into two subsets: $\mathbf{I}$ containing discrete variables and $\mathbf{Y}$ containing continuous ones. The dependencies between individual variables in $\mathbf{X}$ can be represented by a directed acyclic graph (DAG) D with node set V and edge set E. The node set V is partitioned as $V = \Delta \cup \Gamma$ into nodes of discrete (Δ) and continuous (Γ) type. Each discrete variable corresponds to a node in Δ and each continuous variable to a node in Γ. The distribution of a variable X_v at node v only depends on variables $\mathbf{X}_{pa(v)}$ at parent nodes $pa(v)$. Thus, the joint density $p(\mathbf{x})$ decomposes as

$$p(\mathbf{x}) = p(\mathbf{i}, \mathbf{y}) = p(\mathbf{i})p(\mathbf{y}|\mathbf{i}) \\ = \prod_{\delta \in \Delta} p(i_\delta | \mathbf{i}_{pa(\delta)}) \cdot \prod_{\gamma \in \Gamma} p(y_\gamma | \mathbf{y}_{pa(\gamma)}, \mathbf{i}_{pa(\gamma)}). \quad (8)$$

The discrete part, $p(\mathbf{i})$, is given by an unrestricted discrete distribution. The distribution of continuous random variables given discrete variables, $p(\mathbf{y}|\mathbf{i})$, is multivariate normal with mean and covariance matrix depending on the configuration of discrete variables. Since discrete variables do not depend on continuous variables, the DAG D contains no edges from nodes in Γ to nodes in Δ.

For discrete nodes, the situation in CG networks is exactly the same as in the pure case discussed in Section 2: The distribution of $I_\delta|\mathbf{i}_{pa(\delta)}$ is multinomial and parametrized by θ_δ. Compared to the purely Gaussian case treated in Section 2, we have for Gaussian nodes in CG networks an additional dependency on discrete parents. This dependency shows in the regression coefficients and the variance, which now not only depend on the node, but also on the state of the discrete parents:

$$Y_\gamma \mid \beta_{\mathbf{i}_{pa(\gamma)}}, \sigma^2_{\mathbf{i}_{pa(\gamma)}} \sim \text{N}(\mathbf{Y}^\top_{pa(\gamma)} \beta_{\mathbf{i}_{pa(\gamma)}},\ \sigma^2_{\mathbf{i}_{pa(\gamma)}}). \quad (9)$$

As a prior distribution we again take the conjugate normal-inverse-χ^2 distribution as in Eq. 3. For further details on CG networks we refer to (Lauritzen, 1996; Bøttcher, 2004).

3.2 Learning from interventional and non-interventional data

Assuming an uniform prior over network structures D, the central quantity to be calculated is the *marginal likelihood* $p(d|D)$ (Heckerman et al., 1995). In the case of only one type of data it can be written as

$$p(d|D) \ = \ \int_\Theta p(d|D,\theta) p(\theta|D) \ \mathrm{d}\theta. \quad (10)$$

Here $p(\theta|D)$ is the prior on the parameters θ of the LPDs. If the dataset contains both interventional and non-interventional cases, the basic idea is to choose parameter priors locally for each node as in Eq. 6 and 7 according to whether a variable was intervened in a certain case or not. We will see that this strategy effectively leads to a local split of the marginal likelihood into an interventional and a non-interventional part.

3.2.1 A family-wise view of marginal likelihood

To compute the marginal likelihood of CG networks on interventional and non-interventional data, we rewrite Eq. 10 in terms of single nodes such that the theory of (soft) pushing from Section 2 can be used. In the computation we will use the following technical utilities:

1. The dataset d consists of N cases $\mathbf{x}^1, \ldots, \mathbf{x}^N$, which are sampled independently. Thus we can write $p(d|D,\theta)$ as a product over all single case likelihoods $p(\mathbf{x}^c|D,\theta)$, $c = 1, \ldots, N$.

2. The joint density $p(\mathbf{x})$ factors according to the DAG D as in Eq. 8. Thus for each case $\mathbf{x}^c$ we can write $p(\mathbf{x}^c|D,\theta)$ as a product over node contributions $p(x_v^c|\mathbf{x}_{pa(v)}^c,\theta_v)$ for all $v \in V$.

3. We assume *parameter independence*: the parameters associated with one variable are independent of the parameters associated with other variables, and the parameters are independent for each configuration of the discrete parents (Heckerman et al., 1995) This allows us to decompose the prior $p(\theta|D)$ in Eq. 10 into node-wise priors $p(\theta_{v|\mathbf{i}_{pa(v)}}|D)$ for a given parent configuration $\mathbf{i}_{pa(v)}$.

4. All interventions are soft pushing. For a given node, intervention strength and target state stay the same in all cases in the data, but of course different nodes may have different pushing strengths and target values. This constraint just helps us to keep the following formulas simple and can easily be dropped.

These four assumptions allow a family-wise view of the marginal likelihood. Before we present it in a formula, it will be helpful to introduce a *batch notation*. In CG networks, the parameters of the LPD at a certain node depend only on the configuration of discrete parents. This holds for both discrete and Gaussian nodes. Thus, when evaluating the likelihood of data at a certain node, it is reasonable to collect all cases in a batch, which correspond to the same parent configuration:

$$\begin{aligned} &p(d|D,\theta) \\ &= \prod_{c\in d} p(\mathbf{x}^c|D,\theta) \;=\; \prod_{c\in d}\prod_{v\in V} p(x_v^c|\mathbf{x}_{pa(v)}^c,\theta_v) \\ &= \prod_{v\in V}\prod_{\mathbf{i}_{pa(v)}}\prod_{c:\mathbf{i}_{pa(v)}^c=\mathbf{i}_{pa(v)}} p(x_v^c|\mathbf{i}_{pa(v)}^c,\mathbf{y}_{pa(v)},\theta_v) \end{aligned} \tag{11}$$

The last formula is somewhat technical: If the node v is discrete, then $\mathbf{y}_{pa(v)}$ will be empty, and usually not all parent configuration $\mathbf{i}_{pa(v)}$ are found in the data, so some terms of the product will be missing.

For each node we will denote the cases with the same joint parent state by $B_{\mathbf{i}_{pa(v)}}$. When learning with interventional data, we have to distinguish further between observations of a variable which were obtained passively and those that are result of intervention. Thus, for each node v we split the batch $B_{\mathbf{i}_{pa(v)}}$ into one containing all observational cases and one containing the interventional cases:

$$\begin{aligned} B^{obs}_{\mathbf{i}_{pa(v)}} &= \{c \in d \;:\; \mathbf{i}^c_{pa(v)} = \mathbf{i}_{pa(v)} \\ &\qquad \text{and no intervention at } v\}, \\ B^{int}_{\mathbf{i}_{pa(v)}} &= \{c \in d \;:\; \mathbf{i}^c_{pa(v)} = \mathbf{i}_{pa(v)} \\ &\qquad \text{and intervention at } v\}. \end{aligned}$$

If there is more than one type of intervention applied to node v, the batch containing interventional cases has to be split accordingly. Using this notation we can now write down the marginal likelihood for CG networks in terms of single nodes and parents:

$$\begin{aligned} &p(d|D) = \\ &\prod_{v\in V}\prod_{\mathbf{i}_{pa(v)}}\int_\Theta \prod_{B^{obs}_{\mathbf{i}_{pa(v)}}} p(x_v^o|\mathbf{i}_{pa(v)},\mathbf{y}^o_{pa(v)},\theta_v)p'(\theta_v|D)\;\mathrm{d}\theta_v \;\times \\ &\prod_{v\in V}\prod_{\mathbf{i}_{pa(v)}}\int_\Theta \prod_{B^{int}_{\mathbf{i}_{pa(v)}}} p(x_v^e|\mathbf{i}_{pa(v)},\mathbf{y}^e_{pa(v)},\theta_v)p''(\theta_v|D,w_v)\;\mathrm{d}\theta_v. \end{aligned} \tag{12}$$

At each node, we use distributions and priors as defined in Eq. 6 for discrete nodes and Eq. 7 for Gaussian nodes. The non-interventional prior p' corresponds to $F_v = \emptyset$ and the interventional prior p'' corresponds to F_v equalling some target value. We denoted the intervention strength explicitly in the formula, since we will focus on it further when discussing *probabilistic* soft interventions in Section 4.

Equation 12 consists of an observational and an interventional part. Both can further be split into a discrete and a Gaussian part, so we end up with four terms to consider.

3.2.2 Discrete observational part

To write down the marginal likelihood of discrete observational data, we denote by $n_{i_\delta|\mathbf{i}_{pa(\delta)}}$ the number of times we passively observe $I_\delta = i_\delta$ in batch $B^{obs}_{\mathbf{i}_{pa(\delta)}}$, and by $\alpha_{i_\delta|\mathbf{i}_{pa(\delta)}}$ the corresponding pseudo-counts of the Dirichlet prior. Summation of $\alpha_{i_\delta|\mathbf{i}_{pa(\delta)}}$ and $n_{i_\delta|\mathbf{i}_{pa(\delta)}}$ over all $i_\delta \in \mathcal{I}_\delta$ is abbreviated by $\alpha_{\mathbf{i}_{pa(\delta)}}$ and $n_{\mathbf{i}_{pa(\delta)}}$, respectively. Then, the marginal likelihood of the discrete data d_Δ can be written as

$$\begin{aligned} p(d_\Delta \mid D) = \prod_{\delta\in\Delta}\prod_{\mathbf{i}_{pa(\delta)}} \Bigg(&\frac{\Gamma(\alpha_{\mathbf{i}_{pa(\delta)}})}{\Gamma(\alpha_{\mathbf{i}_{pa(\delta)}} + n_{\mathbf{i}_{pa(\delta)}})} \times \\ &\prod_{i_\delta\in\mathcal{I}_\delta} \frac{\Gamma(\alpha_{i_\delta|\mathbf{i}_{pa(\delta)}} + n_{i_\delta|\mathbf{i}_{pa(\delta)}})}{\Gamma(\alpha_{i_\delta|\mathbf{i}_{pa(\delta)}})} \Bigg), \end{aligned} \tag{13}$$

This result was first obtained by Cooper and Herskovits (1992) and is further discussed in (Heckerman et al., 1995).

3.2.3 Discrete interventional part

Since interventions are just changes in the prior, the marginal likelihood of the interventional part of discrete data is of the same form as Eq. 13. The prior parameters $\alpha_{i_\delta|\mathbf{i}_{pa(\delta)}}$ are exchanged by $\alpha'_{i_\delta|\mathbf{i}_{pa(\delta)}} =$

$\mathcal{P}(\alpha_{i_\delta|\mathbf{i}_{pa(\delta)}}, w_\delta, j)$ as given by Eq. 2, and the counts $n_{i_\delta|\mathbf{i}_{pa(\delta)}}$ are exchanged by $n'_{i_\delta|\mathbf{i}_{pa(\delta)}}$ taken from batch $B^{int}_{\mathbf{i}_{pa(\delta)}}$.

In the limit $w_\delta \to \infty$ this part converges to one and vanishs from the overall marginal likelihood $p(d|D)$. This special case was already shown in (Cooper and Yoo, 1999; Tian and Pearl, 2001).

3.2.4 Gaussian observational part

All cases y_γ in batch $B^{obs}_{\mathbf{i}_{pa(\gamma)}}$ are sampled independently from a normal distribution with fixed parameters. If we gather them in a vector $\mathbf{y}_\gamma$ and the corresponding states of continuous parents as rows in a matrix $\mathbf{P}_\gamma$, we yield the standard regression scenario

$$\mathbf{y}_\gamma \sim \mathrm{N}(\mathbf{P}_\gamma \beta_\gamma, \sigma^2_\gamma \mathbf{I}), \tag{14}$$

where $\mathbf{I}$ is the identity matrix. As a prior distribution we choose normal-inverse-χ^2 as shown in Eq. 3. Marginalizing with respect to β_γ and σ^2_γ yields a multivariate t-distribution of dimension $|B^{obs}_{\mathbf{i}_{pa(\gamma)}}|$, with location vector $\mathbf{Pm}$, scale matrix $s(\mathbf{I} + \mathbf{PM}^{-1}\mathbf{P}^\top)$, and ν degrees of freedom. The density function can be found in many textbooks (e.g. Gelman et al., 1996).

When using data from different batches, every parameter above carries an index "$\mathbf{i}_{pa(\gamma)}$" indicating that it depends on the state of the discrete parents of the Gaussian node γ. Multiplying t-densities for all nodes and configurations of discrete parents—the outer double-product in Eq. (12)—yields the marginal likelihood of the Gaussian part. See Bøttcher (2004) for details.

3.2.5 Gaussian interventional part

Here we consider cases y_γ in batch $B^{int}_{\mathbf{i}_{pa(\gamma)}}$. We collect them in a vector and can again write a regression model like in Eq. 14. The difference to the observational Gaussian case lies in the prior parameters. They are now given by Eq. 4. The result of marginalization is again a t-density with parameters as above, just $\mathbf{m}, s$ are exchanged by $(\mathbf{m}', s') = \mathcal{P}((\mathbf{m}, s), w_\gamma, k)$. The Gaussian interventional part is then given by a product of such t-densities over nodes and discrete parent configurations.

If we use the hard intervention prior in Eq. 5 instead, the Gaussian interventional part integrates to one and vanishs from the marginal likelihood in Eq. (12). This is the extension of the results in (Cooper and Yoo, 1999) to Gaussian networks.

4 Probabilistic soft interventions

In Section 2 we introduced the pushing operator $\mathcal{P}(\cdot, w_v, t_v)$ to model a soft intervention at a discrete or Gaussian node v. The intervention strength w_v is a parameter, which has to be chosen before network learning. There are several possibilities, how to do it:

- If there is solid experimental experience on how powerful interventions are, this can be reflected in an appropriate choice of w_v. An obvious problem is that w_v needs to be determined on a scale that is compatible with the Bayesian network model.
- If there is prior knowledge on parts of the network topology, the parameter w_v can be tuned until the result of network learning fits the prior knowledge.

Note again that by the parametrization of pushing given in Section 2, the pushing strengths for discrete and Gaussian nodes live on different scales and have to be calibrated separately.

However, a closer inspection of the biological experiments, which motivated the theory of soft pushing interventions, suggests to treat the intervention strength w_v as a random variable: In gene silencing an inhibiting molecule (a double-stranded RNA in case of RNAi) is introduced into the cell. This usually works in a high percentage of affected cells. In the case of success, the inhibitor still has to spread throughout the cell to silence the target gene. This diffusion process is stochastic and consequently causes experimental variance in the strength of the silencing effect.

These observations suggest to assign a prior distribution $p(w_v)$ to the intervention strength. That is, we drop the assumption of having one intervention strength in all cases, but instead average over possible values of w_v. For simplicity we assume there is only a limited number of possible values of w_v, say, $w_v^{(1)}, \ldots, w_v^{(k)}$, with an arbitrary discrete distribution assigned to them. Then we can express our inability to control the pushing strength in the experiment deterministically by using a mixed prior of the form:

$$p(\theta_v|D) = \sum_{i=1}^{k} q_k \, p(\theta_v|D, w_v^{(k)}). \tag{15}$$

Here, the mixture coefficients $q_k = p(w_v^{(k)})$ are the prior probabilities of each possible pushing strength. The terms $p(\theta_v|D, w_v^{(k)})$ correspond to Dirichlet densities in the discrete case and Normal-inverse-χ^2 densities in the Gaussian case. In RNAi experiments, $w_v^{(1)}, \ldots, w_v^{(k)}$ can be estimated from the empirical distribution of measured RNA degradation efficiencies in repeated assays.

Mixed priors as in Eq. 15 are often used in biological sequence analysis to express prior knowledge which is

not easily forced into a single distribution. See (Durbin et al., 1998) for details.

If we substitute the prior $p''(\theta_v|D, w_v)$ in the interventional part of Eq. 12 with the mixture prior in Eq. 15, the marginal likelihood of a family of nodes is a mixture of marginal likelihoods corresponding to certain values $w_v^{(k)}$ weighted by mixture coefficients q_k.

5 Conclusion

Our work extends structure learning from interventional data into two directions: from learning discrete networks to learning mixed networks and from learning with hard interventions to learning with soft interventions.

Soft interventions are focussed on a specific target value of the variable of interest and concentrate the local probability distribution there. We proposed parametrizations for pushing discrete and continuous variables using Dirichlet and Normal-inverse-χ^2 priors, respectively.

We computed the marginal likelihood of CG networks for data containing both observational and (soft) interventional cases. In Bayesian structure learning, the marginal likelihood is the key quantity to compute from data. Using it (and possibly a prior over network structures) as a scoring function, we can start model search over possible network structures. For networks with more than 5 nodes, exhaustive search becomes infeasible; often used search heuristics include hill climbing or MCMC methods. For a survey see Heckerman et al. (1995) and references therein.

Since in biological settings the pushing strength is unknown we proposed using a mixture hyperprior on it, resulting in a mixture marginal likelihood. This makes the score for each network more time-consuming to compute. Searching in the space of DAGs may become infeasible even with quick search heuristics. But in applications there is often a large amount of biological prior knowledge, which limits the number of pathway candidates from the beginning. When learning network structure we usually don't have to optimize the score over the space of all possible DAGs but are limited to a few candidate networks, which are to be compared. This corresponds to a very rigid structure prior.

Due to measurement error or noise inherent in the observed system it may often happen that a variable, at which an intervention took place, is observed in a state different from the target state. In the hard intervention framework, a single observation of this kind results in a marginal likelihood of zero. Modeling interventions as soft pushing mends this problem and makes structure learning more robust against noise. This is a central benefit of our approach.

Acknowledgements

This work was done within the context of the Berlin Center for Genome Based Bioinformatics (BCB), part of the German National Genome Research Network (NGFN) and supported by BMBF grants 031U109C and 03U117. We thank Dennis Kostka as well as the rest of CMB for helpful discussions.

References

B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*. Garland Science, New York, 4 edition, 2002.

S. G. Bøttcher. *Learning Bayesian Networks with Mixed Variables*. PhD thesis, Aalborg University, Denmark, 2004.

G. F. Cooper. A Bayesian Method for Causal Modeling and Discovery Under Selection. In *Proceedings of UAI 16*, pages 98–106, 2000.

G. F. Cooper and E. Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning*, 9:309–347, 1992.

G. F. Cooper and C. Yoo. Causal discovery from a mixture of experimental and observational data. In *Proceedings of UAI 15*, pages 116–125, 1999.

R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis*. Cambridge University Press, 1998.

N. Friedman. Inferring Cellular Networks Using Probabilistic Graphical Models. *Science*, 303(5659):799–805, 2004.

D. Geiger and D. Heckerman. Learning Gaussian Networks. In *Proceedings of UAI 10*, pages 235–243, 1994.

A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman and Hall-CRC, 1996.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning*, 20(3):197–243, Sep. 1995.

S. L. Lauritzen. *Graphical Models*. Clarendon Press, Oxford, 1996.

S. L. Lauritzen. Causal Inference from Graphical Models. In: O. E. Barndorff-Nielsen, D. R. Cox, and C. Klüppelberg (eds.) *Complex Stochastic Systems*. Chapman and Hall, London, 2000.

K. P. Murphy. Active Learning of Causal Bayes Net Structure, *Tech. Rep., UC Berkeley*, 2001.

J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, Cambridge, 2000.

D. Pe'er, A. Regev, G. Elidan, and N. Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 17(90001):S215–S224, 2001.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, Cambridge, MA, second edition, 2000.

H. Steck and T. S. Jaakkola. Unsupervised active learning in large domains. In *Proceedings of UAI 18*, pages 469–476, 2002.

J. Tian and J. Pearl. Causal discovery from changes: a Bayesian approach. In *Proceedings of UAI 17*, pages 512–521, 2001.

S. Tong and D. Koller. Active Learning for Structure in Bayesian Networks. In *Proceedings of IJCAI 17*, 2001.

T. S. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Proceedings of UAI 6*, pages 255–268, 1990.

C. Yoo and G. F. Cooper. An evaluation of a system that recommends microarray experiments to perform to discover gene-regulation pathways. *Journal Artificial Intelligence in Medicine*, 2003.

C. Yoo, V. Thorsson, and G. F. Cooper. Discovery of causal relationships in a generegulation pathway from a mixture of experimental and oberservational DNA microarray data. In *Proceedings of PSB 7*, pages 498-509, 2002.

Unsupervised Learning with Non-Ignorable Missing Data

Benjamin M. Marlin, Sam T. Roweis, Richard S. Zemel
Department of Computer Science
University of Toronto
Toronto, Ontario

Abstract

In this paper we explore the topic of unsupervised learning in the presence of non-ignorable missing data with an unknown missing data mechanism. We discuss several classes of missing data mechanisms for categorical data and develop learning and inference methods for two specific models. We present empirical results using synthetic data which show that these algorithms can recover both the unknown selection model parameters and the underlying data model parameters to a high degree of accuracy. We also apply the algorithms to real data from the domain of collaborative filtering, and report initial results.

1 Introduction

In large, real world data sets (such as those commonly used for machine learning research), the presence of a certain amount of missing data is inevitable. Probabilistic methods offer a natural framework for dealing with missing data, and there is a large body of work devoted to statistical analysis in this setting.

There are two important classes of missing data: missing data that is ignorable, and missing data that is non-ignorable. Ignorable missing data includes data that is missing completely at random (MCAR), and data that is missing at random (MAR). Intuitively, missing data is ignorable if the probability of observing a data item is independent of the value of that data item. Conversely, missing data is non-ignorable if the probability of observing a data item is dependent on the value of that data item.

The majority of statistical literature deals with the case where missing data is missing at random. However, there are several important cases where the missing at random assumption fails to hold. Well studied examples from statistics include non-response in surveys, panel data studies, and longitudinal studies. In surveys non-ignorable missing data often results from asking questions about income where the probability of non-response has been found to vary according to the income of the respondent. In such cases computing statistics like average income without taking the missing data mechanism into account will result in a biased estimator.

A much more complex domain where missing data may be non-ignorable is rating-based collaborative filtering [5]. The data in this domain often comes from recommender systems where users rate different items and receive recommendations about new items they might like. When a user is free to chose which items they rate, we hypothesize that many users will exhibit a bias toward rating items they like (and perhaps a few they strongly dislike). Thus the probability of observing a rating for a given item will depend on the user's rating for that item, and the missing ratings will not be missing at random. The best known methods for predicting user ratings are based on using unsupervised learning techniques to estimate the parameters of a probabilistic model over rating profiles. Just as in the simple mean income estimation problem, the model parameter estimates will be biased in the presence of non-ignorable missing data.

In this paper we consider the general problem of learning latent variable models in the presence of non-ignorable missing data with an unknown missing data mechanism. We present learning methods based on the Expectation Maximization (EM) algorithm for several different models. We present empirical results on several synthetic data sets showing that the learning procedure recovers the data and selection model parameters to a high degree of accuracy. We also present interesting results on real data from the collaborative filtering domain.

$$\mathcal{L}(\theta|Y^{obs}) = \log f(Y^{obs}|\theta) = \log \int f(Y^{obs}, Y^{mis}|\theta) dY^{mis} \quad (1)$$

$$\mathcal{L}(\theta, \mu|Y^{obs}, R) = \log f(Y^{obs}, R|\theta, \mu) = \log \int f(R|Y^{obs}, Y^{mis}, \mu) f(Y^{obs}, Y^{mis}|\theta) dY^{mis} \quad (2)$$

$$\mathcal{L}_{\text{MAR}}(\theta, \mu|Y^{obs}, R) = \log f(R|Y^{obs}, \mu) + \log \int f(Y^{obs}, Y^{mis}|\theta) dY^{mis} = \mathcal{L}(\theta|Y^{obs}) + \mathcal{L}(\mu|Y^{obs}, R) \quad (3)$$

2 Non-Ignorable Missing Data Theory

We begin with technical definitions of ignorable and non-ignorable missing data due to Little and Rubin, and review the theory of maximum likelihood estimation with non-ignorable data.

Let Y denote a complete data set and let Y^{obs} and Y^{mis} be the observed and missing elements of Y. Let R be a matrix of response indicators where $R_{ij} = 1$ if Y_{ij} is observed and 0 otherwise. We define $f(Y, R|\theta, \mu) = f(Y|\theta) f(R|Y, \mu)$ to be the joint distribution over the data and response indicators. We refer to $f(Y|\theta)$ as the *data model* and $f(R|Y, \mu)$ as the *selection model*.

In the presence of missing data the correct maximum likelihood inference procedure is to maximize the full data likelihood $\mathcal{L}(\theta, \mu|Y^{obs}, R) = \log f(Y^{obs}, R|\theta, \mu)$ shown in equation 2 as opposed to the observed data log likelihood shown in equation 1. The only case where it is acceptable to rely on the observed data likelihood is when the missing at random (MAR) condition holds. The MAR condition is satisfied when $f(R|Y^{obs}, Y^{mis}, \mu) = f(R|Y^{obs}, \mu)$ for all μ, and μ and θ are distinct parameters. If we suppose that the MAR condition holds we find that the full data log likelihood and the observed data log likelihood will give identical inferences for θ, as shown in equation 3.

When missing data is missing not at random (MNAR) this convenient result does not hold, and maximum likelihood estimation of the data model parameters θ based only on the observed likelihood will be biased. To obtain correct maximum likelihood estimates of the data model parameters a selection model is needed along with the data model [3, p. 218]. In most cases the parameters of the selection model will also be unknown. Fortunately the parameters of the combined data and selection model can be estimated simultaneously by maximizing the full data log likelihood using the standard EM algorithm.

3 Non-Ignorable Missing Data Models

Suppose we are given a data set containing N data vectors $\mathbf{y}_n$, each of length M. The value of each element y_{mn} is categorical, and is drawn from the set of possible values $\{1, ..., V\}$. We will assume that the $\mathbf{y}_n$ are independent, that any element y_{mn} may be unobserved, and that the selection process is unknown.

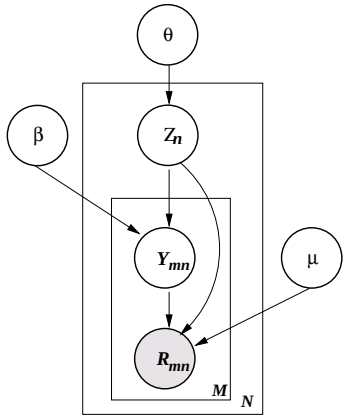


Figure 1: Combined data and selection model.

To specify a model for non-ignorable missing data we must choose a data model and a selection model. We choose a multinomial mixture model for the data model. This is a simple model for categorical data that has proved to be quite robust in the collaborative filtering domain [5]. Several other models for categorical data could also be used including the aspect model [2] and the URP model [4].

The more interesting choice is the choice of a selection model. In general, the probability that a particular variable is observed can depend on any other variable in the data model. Learning such an unstructured model would be intractable, so we must assume additional independence structure. To begin with, we assume that the probability of being observed is independent for each component of the data vector; this is formalized in equation 4. (Analogous simplifying assumptions have been used previously in similar discrete, non-ignorable contexts [6].)

$$P(\mathbf{r}|\mathbf{y}, z) = \prod_{m=1}^{M} \mu_{y_m mz}{}^{(r_m)} (1 - \mu_{y_m mz})^{(1-r_m)} \quad (4)$$

If we represent the Bernoulli observation parameters $P(r_m = 1|y_m = v, Z = z) = \mu_{vmz}$ using conditional

probability tables and combine this selection model with the multinomial mixture data model, we obtain a simple, tractable model for non-ignorable missing data. We call this the *CPT-vmz* model since the μ_{vmz} probabilities are represented by conditional probability tables (CPTs), and the selection probability depends on the settings of the variables v, m, and z. This combined data and selection model is depicted graphically in figure 1.

The variables in the model are the latent variable Z_n, the data variables Y_{mn}, and the corresponding response indicators R_{mn}. Recall that m indexes dimensions of the data vectors, and n indexes data cases. We suppress the data case index for simplicity when it is clear that we are referring to a single, generic data case. The parameters are the prior mixing proportions θ, the component distributions , and the selection model parameters μ. To generate data from the combined multinomial mixture data model and *CPT-vmz* selection model we begin by sampling a state z of the latent variable Z according to $P(Z = z|\theta) = \theta_z$. We then sample a value v for each element y_m according to $P(y_m = v|Z = z,) =$ $_{vmz}$. This is simply the standard generative process for the multinomial mixture model. Next, for each element y_m, we sample a response indicator variable according to the selection probability $P(r_m = r|y_m = v, M = m, Z = z, \mu) = \mu_{vmz}$. We then discard the values of y_m for which $r_m = 0$.

Depending on the type of selection effect that is present in the data, it may be desirable to further constrain the selection model. Imposing independence and factorization conditions on the Bernoulli probabilities μ_{vmz} results in a range of selection models with different properties. In this paper we concentrate on two models in particular: a model we call *CPT-v* that makes the additional independence assumption $P(r_m|y_m, m, z) = P(r_m|y_m)$, and a model we call *LOGIT-v,mz* which proposes a functional decomposition of $P(r_m|y_m, m, z)$ based on the logistic function.

3.1 The *CPT-v* Model

While the *CPT-vmz* model is highly flexible and can model a range of very different selection effects, this flexibility may result in over fitting on many data sets. By contrast the *CPT-v* model asserts that only the value of a random variable affects its chance of being observed or missing. The *CPT-v* model is highly constrained, but this makes it appealing when we have limited observations and cannot robustly fit the full *CPT-vmz* model. Examples of the type of effects this model captures are "mostly high values are observed", or "only extreme values are observed". However, it can not efficiently represent effect of the type "data item m is observed almost always." As we will see, the strict assumptions of the *CPT-v* model can cause problems during model fitting if the data contains strong item-based effects.

Algorithm 1 Expectation Maximization Algorithm for the *CPT-v* model

E-Step:

$$\lambda_{vmzn} \leftarrow (\delta(y_{mn}, v)\mu_v \ _{vmz})^{r_{mn}}((1-\mu_v) \ _{vmz})^{1-r_{mn}}$$
$$\gamma_{mzn} \leftarrow \sum_{v=1}^{V} \lambda_{vmzn}$$
$$\phi_{zn} \leftarrow \frac{\theta_{z_n} \prod_{m=1}^{M} \gamma_{mzn}}{\sum_{z=1}^{K} \theta_{z'} \prod_{m=1}^{M} \gamma_{mzn}}$$

M-Step:

$$\theta_z \leftarrow \frac{\sum_{n=1}^{N} \phi_{zn}}{\sum_{n=1}^{N}\sum_{z=1}^{K} \phi_{zn}}$$
$$_{vmz} \leftarrow \frac{\sum_{n=1}^{N} \phi_{zn}\lambda_{vmzn}/\gamma_{mzn}}{\sum_{n=1}^{N} \phi_{zn}}$$
$$\mu_v \leftarrow \frac{\sum_{n=1}^{N}\sum_{z=1}^{K} \phi_{zn}\sum_{m=1}^{M} r_{mn}\lambda_{vmzn}/\gamma_{mzn}}{\sum_{n=1}^{N}\sum_{z=1}^{K} \phi_{zn}\sum_{m=1}^{M} \lambda_{vmzn}/\gamma_{mzn}}$$

Equation 5 gives the full data likelihood for the *CPT-v* model. We define the intermediate variables λ_{vmzn} and γ_{mzn} in equations 6 and 7.

$$\mathcal{L}(\theta, \ , \mu|[\mathbf{y}]^{obs}, [\mathbf{r}]) = \sum_{n=1}^{N} \log \sum_{z=1}^{K} \theta_z \prod_{m=1}^{M} \gamma_{mzn} \quad (5)$$

$$\lambda_{vmzn} = (\delta(y_{mn}, v)\mu_v)^{r_{mn}}(1-\mu_v)^{1-r_{mn}} \ _{vmz} \quad (6)$$

$$\gamma_{mzn} = \sum_{v=1}^{V} \lambda_{vmzn} \quad (7)$$

The posterior distribution over settings of the latent variable Z is given in equation 8 using the intermediate variables. Of course, the intractable, full posterior over both $\mathbf{y}_n^{mis}$, and Z_n is never needed during learning.

$$P(z|\mathbf{y}_n^{obs}, \mathbf{r}_n) = \phi_{zn} = \frac{\theta_z \prod_{m=1}^{M} \gamma_{mzn}}{\sum_{z=1}^{K} \theta_{z'} \prod_{m=1}^{M} \gamma_{mzn}} \quad (8)$$

Expectation maximization updates can now be found by maximizing the expected complete log likelihood with respect to the data model and selection model parameters. We show the EM algorithm for the *CPT-v* model in algorithm 1.

3.2 The *LOGIT-v,mz* Model

The *CPT-v* model makes the very strong assumption that a single value-based selection effect is responsible for generating all missing data. We would like to allow different effects for different data items m, as well as allowing the setting of the latent variable z to influence the missing data process. As a modeling choice of intermediate complexity, we propose the *LOGIT* family of selection models. The main feature of *LOGIT*

models is the assumption that the selection model parameters μ_{vmz} result from the interaction of multiple lower dimensional factors. In particular, these models allow all of v, m, z to influence the probability of a data element being missing, but constrain the effects to a particular functional family.

In the case of *LOGIT-v,mz* two factors are proposed. One factor σ_v models a value-based effect, while the other factor ω_{mz} models a joint element index/latent variable effect. This latter effect can include factors that are item-specific (a given data item m can have its own probability of being missing), and latent variable-specific (each mixture component z generates its own pattern of missing data). The values of these factors can be arbitrary real numbers and they are combined to obtain the selection probabilities through the logistic function as seen in equation 9. This parameterization was selected because it is more flexible than simple factorizations, such as a product of Bernoulli probabilities. Suppose a data set contains strong value-based selection effects for most data items, but the values for one data items are always observed regardless of their values. *LOGIT-v,mz* can account for this by setting ω_{mz} to a large value. The logistic combination rule then allows ω_{mz} to override σ_v and produce a selection probability of 1 for just this data item. In a product of distributions decomposition this simply is not possible. As we will later see, this flexibility is needed for modeling "blockbuster" effects in the collaborative filtering domain.

$$\mu_{vmz} = \frac{1}{1 + \exp^{-(\sigma_v + \omega_{mz})}} \tag{9}$$

Given values for the selection model parameters σ_v and ω_{mz}, we can compute the complete set of selection probability parameters μ_{vmz} according to equation 9. If we then redefine the intermediate variable λ_{vmzn} according to equation 10, the full likelihood and the posterior over the latent variable have exactly the same form for the *LOGIT-v,mz* model as they do for the *CPT-v* model.

$$\lambda_{vmzn} = (\delta(y_{mn}, v)\mu_{vmz})^{r_{mn}}(1 - \mu_{vmz})^{1-r_{mn}} \quad _{vmz} \tag{10}$$

Unlike the *CPT-v* case, closed form selection model parameter updates cannot be found for *LOGIT-v,mz*. Instead, numerical optimization methods must be used to adapt these parameters. We sketch a suitable EM algorithm for the *LOGIT-v,mz* model in algorithm 2. Note that to ensure the full likelihood is non-decreasing, line search must be used to determine acceptable step sizes at each iteration.

Algorithm 2 Expectation Maximization Algorithm for the *LOGIT-v,mz* model

E-Step:

$$\mu_{vmz} \leftarrow \frac{1}{1+\exp^{-(\sigma_v+\omega_{mz})}}$$

$$\lambda_{vmzn} \leftarrow (\delta(y_{mn}, v)\mu_{vmz})^{r_{mn}}(1 - \mu_{vmz})^{1-r_{mn}}\beta_{vmz}$$

$$\gamma_{mzn} \leftarrow \sum_{v=1}^{V} \lambda_{vmzn}$$

$$\phi_{zn} \leftarrow \frac{\theta_{z_n} \prod_{m=1}^{M} \gamma_{mzn}}{\sum_{z=1}^{K} \theta_{z'} \prod_{m=1}^{M} \gamma_{mzn}}$$

M-Step:

$$\theta_z \leftarrow \frac{\sum_{n=1}^{N} \phi_{zn}}{\sum_{n=1}^{N}\sum_{z=1}^{K} \phi_{zn}}$$

$$\beta_{vmz} \leftarrow \frac{\sum_{n=1}^{N} \phi_{zn}\lambda_{vmzn}/\gamma_{mzn}}{\sum_{n=1}^{N} \phi_{zn}}$$

$$\sigma \leftarrow \sigma - \alpha_\sigma \sum_{n=1}^{N}\sum_{z=1}^{K} \phi_{zn} \sum_{m=1}^{M} \delta(y_{mn}, v)(r_{mn} - \mu_{vmz})$$

$$\omega \leftarrow \omega - \alpha_\omega \sum_{n=1}^{N} \phi_{zn} \sum_{v=1}^{V} (r_{mn} - \mu_{vmz})$$

4 Synthetic Data Experiments

Our first goal is to examine whether, in situations where the assumptions underlying them are satisfied, the proposed models are able to recover both the unknown selection mechanism and a correct model of the data. To this end, we generated synthetic data sets patterned after real data sets from the collaborative filtering domain.

4.1 Generating Synthetic Data

We generate complete synthetic data sets according to a hierarchical Bayesian procedure. In particular, we choose a $K = 6$ component multinomial mixture data model with M=100 data variables, and $V = 5$ values per variable. The mixture model parameters are sampled from an appropriate length Dirichlet prior (uniform, strength two). We sample n=5000 data cases from the mixture model to form a single, complete data set.

To generate data that conforms to the *CPT-v* selection model, we created several sets of selection parameters and used these to sample the complete data. For the purpose of these experiments we use a *CPT-v* selection model with a special functional form that allows the strength of the missing data effect to be easily quantified. In particular we let $\mu_v(s) = s(v-3)+0.5$, where s is the parameter that controls the strength of the effect. Note that since the underlying data distribution is uniform across values, any choice of s in the range $0 < s < 0.25$ yields an overall observation rate of 0.5. We create ten sets of observation probabilities by evenly varying the parameter s from 0 to 0.225.

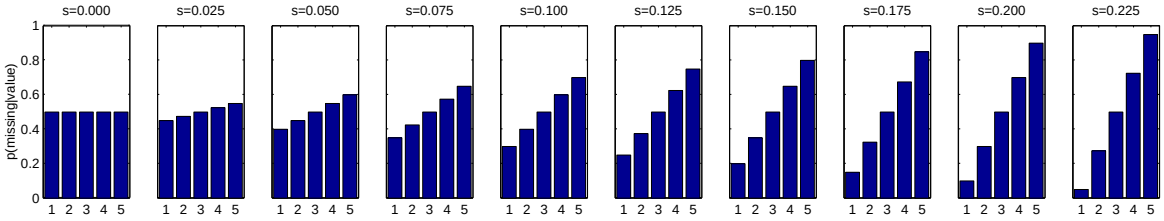


Figure 2: Selection probabilities for the effect $\mu_v(s) = s(v-3)+0.5$. The parameter s controls the strength of the missing data effect. Here we show $\mu_v(s)$ at ten equally spaced values on the interval 0 s 0.225.

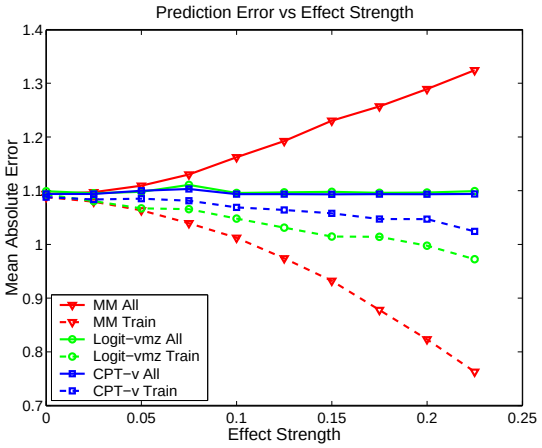


Figure 3: MAE_{All} and MAE_{Tr} versus strength of the *CPT-v* missing data effect for the multinomial mixture, *CPT-v*, and *LOGIT-v,mz* models.

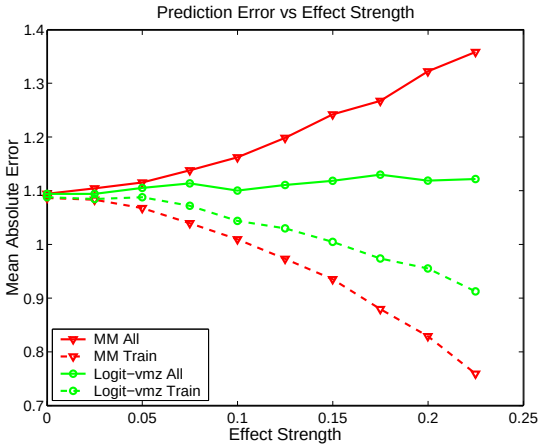


Figure 4: MAE_{All} and MAE_{Tr} versus strength of the *LOGIT-v,mz* missing data effect for the multinomial mixture, and *LOGIT-v,mz* models.

The resulting parameters are shown in figure 2. These ten sets of observation parameters were used to sample ten different training sets from the complete data set.

To generate data that conforms to the *LOGIT-v,mz* model we need to define the selection model parameters σ_v and ω_{mz}. We create 10 different selection models by setting:

$$\sigma_v(s) = \log(\mu_v(s)/(1-\mu_v(s)))$$

$$\omega_{mz} = \log(\ _{mz}/(1- \ _{mz}))$$

The $_{mz}$ values are sampled from a Beta prior. Note that we have chosen these values so that when the logistic function is applied to $\sigma_v(s)$ the result is $\mu_v(s)$, and when it is applied to ω_{mz} the result is $_{mz}$. We compute the corresponding set of selection probabilities $\mu_{vmz}(s)$ and use these to sample ten different training sets.

4.2 Experimental Procedure

The mixture of multinomials model, the *CPT-v* model, and the *LOGIT-v,mz* model were trained until convergence of their respective likelihood functions on all ten of the *CPT-v* training sets, and all ten of the *LOGIT-v,mz* training sets. After training, each model was used to predict the complete data vector $\hat{\mathbf{y}}_n$ for each data case n given the training set (observed) values for that data case. We repeat this training and prediction process three times with different initializations to account for local minima.

In order to judge the performance of each model under increasing missing data effect strength, we use the true and predicted ratings to measure a quantity called the mean absolute error (MAE), which is commonly used as an error measure in the collaborative filtering domain. We measure the MAE restricted to the training data as defined in equation 11, as well as the MAE computed over the complete data set as seen in equation 12. Note that on a real data set we have no choice but to compute the MAE restricted to the observed ratings, which corresponds to equation 11. If a given model accurately learns both the data model and selection model parameters for each setting of the effect strength s, the computed MAE_{All} values should be approximately constant indicating little degradation in performance.

$$MAE_{Tr} = \frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M}\frac{r_{mn}|y_{mn}-\hat{y}_{mn}|}{\sum_{m=1}^{M} r_{mn}} \quad (11)$$

$$MAE_{All} = \frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}|y_{mn}-\hat{y}_{mn}| \quad (12)$$

It is very important to note that under a non-ignorable selection model, the value of MAE_{Tr} as defined by equation 11 is a biased estimate of $E[|y-\hat{y}|]$, were we to sample items uniformly at random. Since the selection model is non-ignorable and also unknown, it is not possible to introduce a correction factor that will allow for an unbiased estimate of $E[|y-\hat{y}|]$ from the training data alone. On the other hand, the value of MAE_{All} as computed by equation 12 is unbiased because it is computed from the complete set of true ratings. However, such a complete set of ratings is not currently available for real data sets.

4.3 Results

In figure 3 we show the results of the synthetic data experiment with the *CPT-v* selection model effect. At $s = 0$, corresponding to the first histogram in figure 2, the selection effect is constant across values v and is thus ignorable. In this case we would expect MAE_{Tr} to be approximately equal to MAE_{All}. In addition we would expect all three models to achieve approximately the same prediction performance since all three are based on an underlying multinomial mixture data model. The experimental results we obtain at $s = 0$ are exactly in line with the theory.

As we increase s we would expect that the value of MAE_{All} would increase for the multinomial mixture model since its parameter estimates are based only on the observed training data. Both the *CPT-v* and *LOGIT-v,mz* models have the capacity to exactly model this selection effect. If these models learn accurate data and selection model parameters then the measured value of MAE_{All} should be approximately constant. A further note is that *LOGIT-v,mz* actually has excess capacity when applied to the *CPT-v* selection effect data, so over fitting may be an issue.

As we see in figure 3 the MAE_{All} curves follow exactly the trends we have predicted for each of the models. However, the MAE_{Tr} curves exhibit an interesting downward trend indicating that error on the training set actually decreases as the missing data effect strength is increased. This is not unexpected in the mixture of multinomials model since it is able to concentrate more of its capacity on fewer rating values and thus achieve lower error on the training data. *CPT-v* and *LOGIT-v,mz* exhibit a slight decrease in error on the training set as the selection strength is increased, but it is not accompanied by a corresponding increase on the complete data set.

In figure 4 we show the results of the synthetic data experiment with the *LOGIT-v,mz* selection effect. The most notable result of this experiment is the fact that the learning procedure for the *CPT-v* model, algorithm 1, converges to a "boundary solution" for the μ_v parameters for all values of the effect strength s. Specifically, at convergence the μ values have the form $\mu_1 = c$, μ_j 1, where c reflects the global sparsity rate and 2 j 5. This appears to indicate that the *CPT-v* model lacks the capacity to model the *LOGIT-v,mz* missing data effect. This failure of the *CPT-v* model may result from the presence of strong item-based effects in the *LOGIT-v,mz* data. For example, suppose an item is always observed regardless of its value. The only way *CPT-v* can explain this is by increasing the values of μ. Of course it cannot increase all the values of μ since it must still explain the fraction of data that is missing. The most likely solution appears to be exactly the boundary solution explained above. This problem may also simply be a failure of the maximum likelihood framework. We plan to explore a Bayesian approach to prediction to determine if the problem actually lies with the model, or the estimation and prediction techniques.

The trends for the mixture of multinomials model are quite similar to the previous case with similar explanations applying. The trends for the *LOGIT-v,mz* model are also similar to the previous case. One slight difference is that the MAE_{All} curve is more noisy and increases somewhat with s. The most likely explanation is an insufficient amount of training data to properly estimate all the ω_{mz} parameters. The previous case is easier for the *LOGIT-v,mz* model in this respect since the *CPT-v* data contains no item or latent variable-based effects.

Perhaps the most important point illustrated by figures 3 and 4 is that estimating the prediction error on the observed data only can be an arbitrarily poor estimate of the error on the complete data set in the presence of a non-ignorable missing data effect. In both graphs we see that the multinomial mixture model attains the lowest error on the observed data, when in fact its true error rate is the highest among the three models.

5 Real Data Experiments

Unfortunately, the task of evaluating a method for unsupervised learning in the presence of non-ignorable

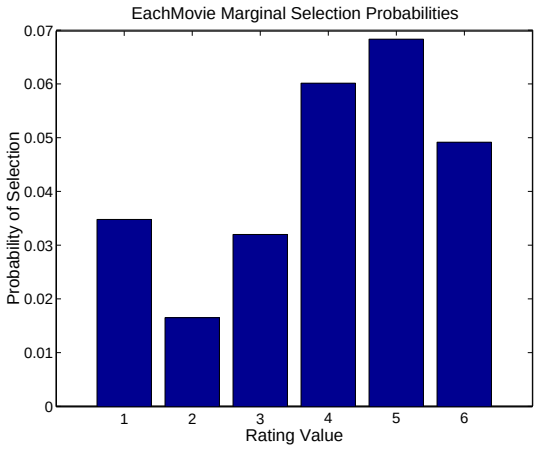


Figure 5: EachMovie marginal selection probabilities $P(r = 1|y = v)$. Computed under a learned *LOGIT-v,mz* model from parameters $\sigma, \omega, \ , \theta$.

Figure 6: EachMovie Full Data Log Likelihood

	Full Data Log Likelihood
LOGIT-v,mz	-8.750368×10^6
Multinomial Mixture	-1.164895×10^7

missing data on real data sets is problematic. Standard evaluation procedures involve measuring prediction error on held out observed data. This hold-out method entails an inherent bias in our case, as it is only possible to evaluate predictions for items that are not missing. That is, unlike the case for synthetic data, evaluating MAE_{All} is impossible for real data. A learning method based on the MAR assumption would be expected to perform at least as well as one based on MNAR on such a biased hold-out experiment: a system can safely ignore any dependence of the item's response on missing values if it is never evaluated on missing items.

We are currently considering a novel interactive data gathering procedure designed to collect exactly the type of data needed to validate the proposed models. The main idea is to conduct an interactive, two stage survey. In the first stage participants would be free to select and rate any number of items they wished. This data would then form the training set, and would likely contain a strong non-ignorable missing data effect. In the second stage of the survey a different set of items would be randomly selected for each participant to rate. Ratings for these randomly sampled items would then form the test set. While this might be difficult for items like movies, it is quite simple for items like music where a clip can be played if the participant is not familiar with a particular randomly chosen item. Since the test data is a randomly chosen set of items, evaluating prediction error on the test set gives an unbiased estimate of the error on the whole data set.

It is still interesting to observe what the proposed models learn on currently available real data sets, even if this results can not be considered conclusive. We note that not all currently available ratings-based collaborative filtering data can be used with the proposed models. The key assumption is that users are free to rate items at will. Any source of data where the users are constrained to rate a sequence of items determined solely by a recommender system violates this assumption. In this case the proposed models would essentially be learning the selection bias caused by the recommender system attempting to predict items it thinks the user will like. While this may actually be an interesting method to test the efficacy of a recommender system, it is not what we intend here.

We chose to apply the *CPT-v* and *LOGIT-v,mz* models to a well known collaborative filtering data set: EachMovie. EachMovie is a collaborative filtering data set collected from a movie recommender system operated by the Compaq Systems Research Center. It contains about 1600 movies and 70000 users. EachMovie is well known to contain a "blockbuster" effect where several movies have been rated by almost all users, while others are rated by just a few users. EachMovie also contains a fairly strong user-based effect: the number of movies rated per user ranges from one to several thousand. EachMovie is widely believed to contain quite a substantial bias toward high rating values. The underlying recommender system used to collect the data also allowed for free user interaction, which satisfies our main requirement.

EachMovie appears too complicated for the *CPT-v* model using maximum likelihood learning. As in the synthetic *LOGIT-v,mz* experiment, *CPT-v* converges to uninformative boundary solutions on EachMovie. On the other hand, *LOGIT-v,mz* appears to converge to parameter estimates that are in very good alignment with the effects that are thought to occur in the data set.

After convergence, we can examine the learned parameters $\sigma, \omega, \ , \theta$ and use them to compute the marginal probability $P(r = 1|y = v)$ as a summary of the learned selection effect (averaged across all items and settings of the latent variable). We show the computed marginal selection probabilities for EachMovie in figure 5. This figure exhibits a definite skew toward high ratings values, although the dip at the highest rating value is somewhat suspect.

While we cannot compute an unbiased estimator of the expected mean absolute error for the data set, we can compute another quantity that will allow us to com-

pare the *LOGIT-v,mz* model with a MAR multinomial mixture model: the full data likelihood. The mixture of multinomials model has no explicit selection model and optimizes only an observed data likelihood as seen in equation 1. However, as we see in equation 3 we can obtain the full data likelihood from the observed data likelihood by accounting for the likelihood of the response indicators. If we suppose a simple MAR scheme with a global observability parameter μ, the log likelihood of the complete set of response indicators is given by $\log(\mu) \sum_n \sum_m r_{mn} + \log(1-\mu) \sum_n \sum_m (1 - r_{mn})$. In this case the maximum likelihood estimator for μ is simply $\mu = \frac{1}{NM} \sum_n \sum_m r_{mn}$.

We show the full data likelihood values for the *LOGIT-v,mz* model and the multinomial mixture model computed for the EachMovie data set in figure 6. We see that *LOGIT-v,mz* obtains a significantly higher full data log likelihood value than the simple MAR multinomial mixture model.

6 Extensions and Future Work

In addition to the research directions mentioned in the previous sections, we are also considering extensions of the proposed framework that will allow us to model a wider range of missing data problems. In the original framework, the binary response value R_m for an item m is determined by the presence of a rating Y_m for that item. We might also decouple these two variables, and allow a response to exist for an item ($R_m = 1$) when the rating is not known. This situation often arises in Web-based systems where we may have information about many items a user has viewed, but a relatively small number of ratings. Only minor changes are required to reformulate the proposed models to handle this type of data.

7 Conclusions

In this paper, we have proposed several probabilistic selection models which treat missing data as a systematic (non-ignorable) rather than random (ignorable) effect. Coupled with a basic discrete latent variable model for user-item data, these selection mechanisms allow us to model data sets in which known (or suspected) response biases exist. We have derived efficient learning and inference algorithms to jointly estimate the data and selection model parameters in an unsupervised way, and verified that these algorithms can recover both the unknown selection model parameters and the underlying data model parameters to a high degree of accuracy under a wide variety of conditions. We have also shown that when an unbiased estimate of their performance is available, our models do substantially better than comparable models which do not account for the missing data mechanism. Finally, we have applied these models to a real world collaborative filtering data set, EachMovie, obtaining initial results that support several "folk beliefs" about the patterns of missing data in this data set.

Acknowledgments

We thank Nathan Srebro for helpful comments and discussions about missing data and collaborative filtering. His comments on an internal presentation of this work were instrumental in revising this paper. We thank Krisztina Filep for reviewing the final draft. Lastly, we thank the Compaq Computer Corporation for the use of the EachMovie data set.

References

[1] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval Journal*, 4(2):133–151, July 2001.

[2] T. Hofmann. Learning What People (Don't) Want. In *Proceedings of the European Conference on Machine Learning (ECML)*, 2001.

[3] R. J. A. Little and D. B. Rubin. *Statistical analysis with missing data*. John Wiley & Sons, Inc., 1987.

[4] B. Marlin. Modeling user rating profiles for collaborative filtering. In *Proceedings of the Seventeenth Annual Conference on Neural Information Processing Systems (NIPS-2003)*, 2003.

[5] B. Marlin. Collaborative filtering: A machine learning perspective. Master's thesis, University of Toronto, January 2004.

[6] E. A. Ramalho and R. J. Smith. Discrete choice models for nonignorable missing data. In *Proceedings of the Econometric Society Fifty-Seventy European Meeting (ESEM'2002)*, 2002.

Regularized spectral learning

Marina Meilă
Department of Statistics
University of Washington
Seattle, WA 98195

Susan Shortreed
Department of Statistics
University of Washington
Seattle, WA 98195

Liang Xu
Department of Mathematics
University of Washington
Seattle, WA 98195

Abstract

Spectral clustering is a technique for finding groups in data consisting of similarities S_{ij} between pairs of points. We approach the problem of learning the similarity as a function of other observed features, in order to optimize spectral clustering results on future data. This paper formulates a new objective for learning in spectral clustering, that balances a clustering accuracy term, the *gap*, and a stability term, the *eigengap* with the later in the role of a regularizer. We derive an algorithm to optimize this objective, and semiautomatic methods to chose the optimal regularization. Preliminary experiments confirm the validity of the approach.

1 Introduction

While there has been much progress in obtaining better spectral clusterings with similarities given or constructed by hand, the problem of automatically estimating the similarities from data has received less attention. This limits the application of spectral clustering to the ability of the domain experts to guess the correct features and their optimal combination for each problem. It makes the results of the clustering algorithm sensitive to the particular function chosen. Moreover, it goes against the grain of many successful approaches in machine learning, which is to consider a large number of possibly irrelevant features, within a regularized setting that lets the data select the few relevant ones.

In contrast to previous work [Meilă and Shi, 2001a, Bach and Jordan, 2004], where the focus was on defining a quality criterion to be optimized w.r.t the parameters on training data, here we take an approach closer to the principles above. We define an objective that balances a term for clustering accuracy on training data with a stability term which acts as a regularizer. While this setting is common to many problems, the specific form of the two terms is particular to spectral clustering.

We start by introducing notation and some basic facts in section 2, we define the learning problem in section 3 and introduce the new objective in 4. Sections 5, 6 present respectively a gradient algorithm for optimizing the criterion and a method for selecting the amount of regularization. Experimental results are in section 7 and 8 concludes the paper.

2 Spectral clustering – notation and background

In spectral clustering, the data is a set of *similarities* S_{ij}, satisfying $S_{ij} = S_{ji} \geq 0$, between pairs of points i, j in a set V, $|V| = n$. The matrix $S = [S_{ij}]_{i,j \in V}$ is called the *similarity matrix*. We denote by

$$D_i \equiv Vol\{i\} = \sum_{j \in V} S_{ij} \tag{1}$$

the *volume* of node $i \in V$ and by D a diagonal matrix formed with $D_i, i \in V$. The volume of a set $A \subseteq V$ is $Vol\ A = \sum_{i \in A} D_i$. W.l.o.g we assume that no node has volume 0.

The random walks view Many properties of spectral clustering are elegantly expressed in terms of the stochastic *transition matrix* P obtained by normalizing the rows of S to sum to 1.

$$P = D^{-1}S \quad \text{or} \quad P_{ij} = S_{ij}/D_i \tag{2}$$

This matrix can be viewed as defining a Markov random walk over V, P_{ij} being the *transition probability* $Pr[i \to j|i]$. The eigenvalues of P are $1 = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq -1$ and the corresponding eigenvectors are $v^1, \ldots v^n$. Note that because $S = DP$ is symmetric, the eigenvalues of P are real and the eigenvectors linearly independent. Define $[\pi_i]_{i \in V}$ by

$$\pi_i = D_i / Vol\ V$$

It is easy to verify that $P^T\pi = \pi$ and thus that π is a *stationary distribution* of the Markov chain. For a set $A \subseteq V$, we denote by $\pi_A = Vol\ A / Vol V$ the probability of A under the stationary distribution.

The MNCut criterion A clustering $\mathcal{C} = \{C_1, \ldots, C_K\}$ is defined as a partition of the set V into the disjoint nonempty sets $C_1, \ldots, C_K$. The *multiway normalized cut (MNCut)* clustering criterion [Meilă, 2002, Yu and Shi, 2003]

$$MNCut(\mathcal{C}) \;=\; \sum_{k=1}^{K} \sum_{k' \neq k} \frac{Cut(C_k, C_{k'})}{Vol\, C_k} \tag{3}$$

where

$$Cut(A, B) \;=\; \sum_{i \in A} \sum_{j \in B} S_{ij} \tag{4}$$

The definition of *MNCut* is best motivated by the Markov random walk view. Define $P_{AB} = Pr[A \to B|A]$ as the probability of the random walk going from set $A \subset V$ to set $B \subset V$ in one step if the current state is in A and the random walk is in its stationary distribution π.

$$P_{AB} \;=\; \frac{\sum_{i \in A, j \in B} \pi_i P_{ij}}{\pi_A} \;=\; \frac{\sum_{i \in A, j \in B} S_{ij}}{Vol\, A} \;=\; \frac{Cut(A, B)}{Vol\, A} \tag{5}$$

It follows that the multiway normalized cut represents the sum of the "out-of-cluster" transition probabilities at the cluster level.

$$MNCut(\mathcal{C}) \;=\; \sum_{k=1}^{K} \sum_{k \neq k'} P_{C_k C_{k'}} \;=\; K - \sum_{k=1}^{K} P_{C_k C_k} \tag{6}$$

If $MNCut(\mathcal{C})$ is small for a certain partition $\mathcal{C}$, then the probabilities of evading C_k, once the walk is in it, is small.

In [Meilă, 2002] it is shown that the $MNCut(\mathcal{C})$ for any clustering $\mathcal{C}$ is lower bounded by a function of the number of clusters $K = |\mathcal{C}|$ and of the eigenvalues of P:

$$MNCut(\mathcal{C}) \;\geq\; K - \sum_{k=1}^{K} \lambda_k(P) \tag{7}$$

We call the non-negative difference between the *MNCut* and its lower bound the *gap*:

$$gap_P(\mathcal{C}) \;=\; MNCut(\mathcal{C}) - K + \sum_{k=1}^{K} \lambda_k(P) \tag{8}$$

One can show [Meilă, 2002] that the gap is 0 iff P has *piecewise constant eigenvectors (PCE)* $v^1, \ldots v^K$ w.r.t $\mathcal{C}$, that is $v_i^k = v_j^k$ for all $k \leq K$ whenever i, j are in the same cluster.

3 The learning problem

We assume that we have a data set of size n, for which the correct clustering $\mathcal{C}^*$ is given. For each pair of data points i, j in the data set we also measure a set of features. The F-dimensional vector of features is denoted by

$$x_{ij} \;=\; [x_{ij,1}\; x_{ij,2}\; \ldots x_{ij,F}]^T \tag{9}$$

The features are *symmetric*, that is $x_{ij} = x_{ji}$ for all i, j. We also assume for now that the features are non-negative $x_{ij,f} \geq 0,\ f = 1, \ldots F$ and that an increase in $x_{ij,f}$ represents a decrease in the similarity between i and j. One can think of the data points as vectors in some F-dimensional space, with the features representing distances between points along the F coordinate axes. Our formulation however is significantly more general, in that it accomodates dissimilarity features that do not come from a Euclidian space representation of the data.

The *similarity* is an (almost everywhere) differentiable function $S(x; \theta)$ that maps a feature vector $x \in R^F$ into a non-negative scalar similarity (e.g $S(x; \theta) = e^{-\theta^T x}$) with $\theta \in R^F$ a vector of parameters. Let

$$S_{ij} \;=\; S(x_{ij}; \theta) \quad \text{for } i, j = 1, \ldots n \tag{10}$$

$$\mathbf{x} \;=\; [x_{ij}]_{i,j=1,\ldots n} \;\in\; R^{n \times n \times F} \tag{11}$$

$$S(\theta) \;\equiv\; S(\mathbf{x}; \theta) \;\equiv\; [S_{ij}]_{i,j=1,\ldots n} \;\in\; R^{n \times n} \tag{12}$$

Here, the letter S denotes both the similarity function $S(x; \theta)$, and the *similarity matrix* $S(\theta)$ for fixed data set $\mathbf{x}$ and parameter vector θ. The matrices obtained from $S(\theta)$ by (1,2) are respectively denoted $D(\theta)$, $P(\theta)$.

We want to learn the optimal parameters θ of the similarity function from a training set including one or more data sets together with their correct clusterings. For simplicity, we assume that we have one data set described by the features $\mathbf{x}$ together with its correct clustering $\mathcal{C}^*$ having K clusters. The generalization to more than one data set is immediate.

This problem was proposed in [Meilă and Shi, 2001a]; there, the authors introduce a target S^* having $S^*_{ij} = 1$ if i, j are in the same cluster and $S^*_{ij} = 0$ otherwise. The parameters are then optimized by making $S(\theta)$ match S^* in a KL-divergence sense that reflects the random walk interpretation of the similarity matrix. This approach worked well on image segmentation data, but is not appropriate for general purpose learning. For any clustering there can be an infinity of S matrices that are perfect for that clustering. Imposing a target S^* of any form will overconstrain the problem and is equivalent to introducing a bias, which may or may not fit the problem at hand.

In [Bach and Jordan, 2004] an angle between subspaces is used as a criterion for learning the parameters of spectral clustering, thus dispensing with the target S^*. This angle is minimized when $P(\theta)$ has PCE for the given clustering. Optimizing this criterion is difficult in practice due to the need to differentiate a function of the eigenvectors of a matrix w.r.t the matrix elements.

Here, we address learning by explicitly enforcing that the learned parameters induce a good clustering on the training data $(\mathbf{x}, \mathcal{C}^*)$ while "extracting as little information from the trainig data as possible" as will be shown in the next section.

4 The objective

Quality of the target clustering In the context of spectral clustering, we can satisfy the first of the above requirements by enforcing the quality of the true clustering $\mathcal{C}^*$ w.r.t $S(\theta)$. The quality of a clustering can be measured by either its *MNCut* or its gap. For a given matrix S, a clustering that minimizes the first also minimizes the second, so the criteria are equivalent. However, if one *learns* S, then the criteria are not equivalent: obtaining a small *MNCut* implies that the off-diagonal blocks of S are nearly 0, while a small gap does not carry such an implication. Hence, the gap puts fewer constraints on θ while still being an indicator of a good clustering, and we choose it as a criterion of clustering quality.

The eigengap and the stability of the optimal clustering To enforce the second requirement, we will maximize the eigengap $\Delta_K = \lambda_K(P(\theta)) - \lambda_{K+1}(P(\theta))$. To motivate this choice, one can recall the fact that a large eigengap in P makes the subspace spanned by $v^1 \dots v^K$ stable to perturbations. The following result, proved in the appendix, gives a direct relationship between spectral clustering and the eigengap.

For two clusterings with $|\mathcal{C}| = |\mathcal{C}'| = K$ we define the distance of $\mathcal{C}$ and $\mathcal{C}'$ by

$$d(\mathcal{C}, \mathcal{C}') \;=\; 1 - \frac{1}{K} \sum_{C_k \in \mathcal{C}} \sum_{C'_k \in \mathcal{C}'} \frac{(VolC_k \cap C'_{k'})^2}{VolC_k VolC'_{k'}} \qquad (13)$$

This distance is symmetric, ranges in [0,1], but is not a metric. Note that $K - 1 - Kd(\mathcal{C}, \mathcal{C}')$ is Pearson's χ^2 function, [Lancaster, 1969] known in statistics as a measure of departure from independence. The distance d is equivalent with a criterion proposed by [Hubert and Arabie, 1985], with the modification that each data point is weighted by its volume D_i. The unweighted d distance was also used in [Bach and Jordan, 2004].

Theorem 1 *Let $\mathcal{C}, \mathcal{C}'$ be two K-way clusterings with $gap(\mathcal{C})$, $gap(\mathcal{C}') \leq \varepsilon < \Delta_K$. Then, $d(\mathcal{C}, \mathcal{C}') < \frac{3\varepsilon}{\Delta_K} = \delta$.*

Corollary 2 *Let $\mathcal{C}$ be a K-way clustering with $gap(\mathcal{C}) \leq \varepsilon < \Delta_K$ and $\mathcal{C}^* = \underset{|\mathcal{C}'|=K}{\operatorname{argmin}}\, MNCut(\mathcal{C}')$. Then $d(\mathcal{C}, \mathcal{C}^*) < \delta$.*

In words, the above theorem and its corollary show that, if we find a clustering with a small enough gap relative to the eigengap, then that clustering is also "stable", in the sense that any other clustering with small gap will necessarily be close to it. If the eigengap is sufficiently large w.r.t to the best attainable gap, then there is essential a unique way of obtaining a good partition in that P. Any two partitions with a small gap have to be close to each other.

The learning criterion Now we define learning in spectral clustering as solving the following optimization problem

$$(\mathcal{P}) \qquad \max\ \Delta_K^2(P(\theta)) \qquad (14)$$

$$\text{s.t. } gap_\theta(\mathcal{C}^*) \;\leq\; \varepsilon \qquad (15)$$

where gap_θ is a short form for $gap_{P(\theta)}$. By simultaneously achieving a small gap for $\mathcal{C}^*$ and a large eigengap for $P(\theta)$, we enforce that $\mathcal{C}^*$ is both a "good" and a "stable" clustering (all other clusterings with small gap are close to $\mathcal{C}^*$); this has been known to predict good generalization performance in other learning settings.

Our formulation is reminiscent of the SVM formulation; we maximize a "stability" penalty, while ensuring that the training data are well clustered. The parameter ε controls the trade-off between the two goals; ε being a gap, its value is bounded in $[0, K]$. Thus, as a rule of thumb for the choice of ε, a value in $[10^{-2\dots-1}, 1]$ should represent a good enough quality for $\mathcal{C}^*$. In section 6 we give a method for semi-automatically selecting its value.

One can also consider other formulations that balance the gap and eigengap. For example, according to theorem 1, a natural optimality criterion would be $J' = gap_\theta(\mathcal{C})/\Delta_K(P(\theta))$. We chose to optimize $(\mathcal{P})$ over J' because the latter contains a division by the eigengap. Since the eigengap is typically a small number computed as the difference of two consecutive eigenvalues, such an operation is unstable numerically.

5 Optimizing the parameters

Here we present the solution to problem $(\mathcal{P})$. Unlike the SVM formulation, in our optimization problem neither objective nor constraints are convex. Therefore, we optimize the parameters by following the gradient, starting from small, positive, random values for the parameters θ.

The constrained optimization problem (14-15) is equivalent with minimizing

$$J_\alpha \;=\; \alpha\, gap_\theta(\mathcal{C}^*) - \Delta_K^2(P(\theta)) \qquad (16)$$

for an α that depends on ε. If ε is known, Lagrange multiplier methods typically find α and θ simultaneously [Bertsekas, 1999]. Here however we choose to take a different approach, that will be described in the next section. For now, we will consider optimizing for θ with a fixed α.

To evaluate the gradient of J_α we write the criterion as the sum of the *MNCut* and a function of the eigenvalues of P. We further recall that [Meilă and Shi, 2001b] the eigenvalues of $P(\theta)$ are equal to the eigenvalues of the symmetric matrix $L(\theta) = D(\theta)^{-1/2} S(\theta) D(\theta)^{-1/2}$. Therefore we drop the θ in S, L, and P to simplifiy notation

$$
\begin{aligned}
J_\alpha(\theta) = & \\
&= \alpha \left[MNCut_P(\mathcal{C}^*) - K + \sum_{k=1}^{K} \lambda_k(P) \right] - [\lambda_K(P) - \lambda_{K+1}(P)]^2 \\
&= \underbrace{-\alpha \sum_{k=1}^{K} \frac{\sum_{i,j \in k} S_{ij}}{\sum_{i \in k} \sum_{j=1}^{n} S_{ij}}}_{J_1(\theta)} \\
&\quad \underbrace{+ \alpha \sum_{k=1}^{K} \lambda_k(L) - [\lambda_K(L) - \lambda_{K+1}(L)]^2}_{J_2(\theta)}
\end{aligned}
$$

The derivative of J_1 w.r.t to θ is straightforward, assuming that the partial derivatives $\frac{\partial S}{\partial \theta}$ can be computed tractably. In the following we show how to obtain the gradient of the second term, which involves the eigenvalues of $L(\theta)$. Evaluating $\frac{\partial L_{ij}}{\partial \theta}$ also presents no problem, so we focus on the derivatives of eigenvalues and of sums of eigenvalues w.r.t to the elements of L.

It is known that for a symmetric matrix L, the derivative of a simple eigenvalue λ w.r.t the matrix elements has the expression

$$\frac{\partial \lambda}{\partial L} = v v^T \tag{17}$$

where v is the eigenvector corresponding to λ. If λ is a multiple eigenvalue, then $\lambda(L)$ is not differentiable at L. In practice, when two of the eigenvalues of $L(\theta)$ are too close, the evaluation of the gradient becomes numerically instable. When optimizing (16), reducing the *MNCut* term has the tendency to push the first K eigenvalues toward 1, thus sending the optimization trajectory into an instability zone. However, it is worth noting the stabilizing effect played by the eigengap term in (16). Enforcing one large eigengap, Δ_K, for this problem, has the effect of enlarging the distances between all of $\lambda_1, \ldots \lambda_{K+1}$. In fact, in our experiments, the simple gradient given by formula (17) is stable for all but the extremely large values of α.

5.1 Computation

Each gradient step comprises an evaluation of $\frac{\partial J_\alpha}{\partial \theta}$ for the descent direction and several evaluations of J_α for the line search. With the features $\mathbf{x}$ given, computing S takes $\mathcal{O}(n^2 F)$ operations, computing the *MNCut* and $\frac{\partial MNCut}{\partial S}$ takes $\mathcal{O}(n^2)$ and evaluating the partial derivatives $\frac{\partial S}{\partial \theta}$, $\frac{\partial L}{\partial \theta}$ requires another $\mathcal{O}(n^2 F)$. To complete the evaluation of J_α and of its gradient, we also need the first $K+1$ eigenvalues and eigenvectors of $L(\theta)$. We compute them with the Matlab `eigs` function that calls an iterative procedure whose time per iteration is nK', where K' is the number of eigenvalues required. In our experiments, we use $K' = \max(2K, K+10)$. Thus the running time per gradient step is $\mathcal{O}(n^2 F + nK')$.

We also use line search along the direction of descent to find the optimal step size at each iteration. The procedure we use is the Armijo rule [Bertsekas, 1999]. This takes a step of size τ the first time when $J_\alpha(\theta) - J_\alpha(\theta - \tau \frac{\partial J_\alpha}{\partial \theta}) > \beta || \frac{\partial J_\alpha}{\partial \theta} ||_2$ with $\beta = 10^{-2}$, otherwise τ is reduced by a factor of 2. We allow up to 30 reductions, but the algorithm can be easily tuned so that in practice most steps are taken after just 1-2 function evaluations. The implementation of the adaptive step size is however not superfluous, as typically at the beginning and at the end of the learning, smaller step sizes are chosen. With the adaptive step size, the gradient descent usually takes less than 100 steps to attain a good set of θ values; attaining convergence once near the optimum is slower, typical of gradient algorithms. A typical method of speeding up the eigenvalue computation in spectral clustering is to use sparse similarity matrices instead of full ones or the Nyström approximataion for eigenvectors. [Fowlkes et al., 2004] These tricks can be also applied to learning spectral clustering, leading to additional time savings.

6 Selecting the regularization parameter

Choosing the amount of regularization is critical for the success of learning. By translating α into an ϵ via the optimization problem $(\mathcal{P})$, one gets a handle on the order of magnitude of ϵ which can be used when there is good prior knowledge about the problem or when a cheap solution is needed. Now we show a simple method for semi-automatically selecting α.

Algorithm SELECTALPHA

1. Choose a set A of α values spanning a reasonable range, e.g $[10^{-2}, 10^2]$
2. For $\alpha \in A$
   (a) Find $\hat{\theta}_\alpha = \operatorname{argmin}_\theta J_\alpha(\theta)$
   (b) Compute $\delta_\alpha = \Delta_K(P(\hat{\theta}_\alpha))$, $g_\alpha = gap_{\hat{\theta}_\alpha}(\mathcal{C}^*)$
3. (Manual) Choose α^* so that g_{α^*} is small, but δ_{α^*} is still reasonably large. In a plot of g_α vs δ_α, the desired α^* will be near the lower right corner of the plot.

Figure 1 shows an example of such a plot. Note that more than one α value may be near the knee of the curve. This algorithm could be refined, for example by a binary search on α, but the hope is that the problem is not so sensitive to the value of the regularizing parameter.

7 Experiments

In this section we provide the details and results of experiments run with the learning algorithm presented here. The similarity is defined as $S(x, \theta) = e^{-\theta^T x}$, with θ assumed positive. The initial weights were chosen to be inversely proportional to the variance of the $x_{ij,l}$. After learning for each prespecified α two optimal α values and

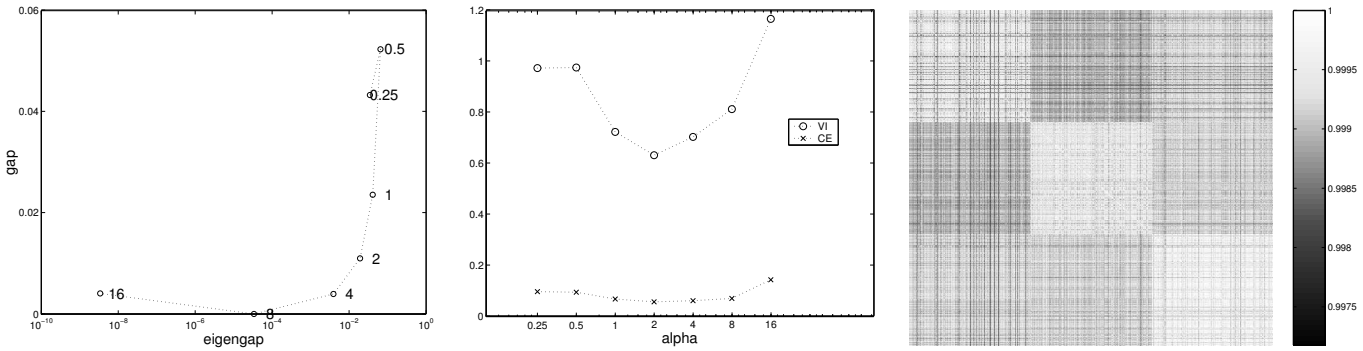


Figure 1: Selecting α on the "letters A,C,I" data set described in section 7: **(a)** The gap vs. eigengap plot on the training data; each point is labeled with its α value. **(b)** Clustering quality for the learned θ's on an independent test set, measured by the variation of information (VI) and the classification error (CE), plotted versus the regularization parameter α. **(c)** The learned similarity matrix S for this dataset. Note that the off-diagonal blocks are non-zero.

corresponding parameters were chosen. The first optimal α was chosen based on the smallest misclassification error on the training set, with ties going to the smallest α, its corresponding parameters will be denoted $\theta_{\alpha,CE}$. The second α was chosen using the select Select Alpha algorithm described in Section 6, and its parameters will be denoted $\theta_{\alpha,SA}$. Selecting a single α is less clear in this second method; in Figure 1a it can be seen that both 2 and 4 are possible candidates for an optimal α. In these situations, it appears that reasonable candidates for α lead to similar clustering results. In order to test each vector of parameters the first K eigenvectors of the corresponding $P(\theta)$ matrix were clustered using k-means, with multiple random and orthonormal initializations; for more information on initializations see [Verma and Meilă, 2003].

7.1 Bull's-Eye

The first set of experiments was run on simulated data, a bull's-eye in two dimensions. The data consist of an inner ring containing approximately 40% of the data points with the remaining data points forming an outer ring. While this data is artificial, large within cluster distances, a small number of neighbors for each point, and a high possibility of over-fitting make learning non-trivial. Also, the meaningful features each taken separately do not correlate well with the clustering, making the high weighting of both meaningful features important in the learning process. We added *Ndim* noisy dimensions to the bull's-eye, which were symmetric random matrices designed to have the same mean as the meaningful features. The weights were learned for a vector of α's, by minimizing (16). The distance metric used for creating the two meaningful dimensions was $x_{ij,l} = |y_{il} - y_{jl}|$. Where y_{i1} is the x-coordinate and y_{i2} is the y-coordinate associated with the i^{th} point. We tested the weights on ten independent samples of 300 data points.

Table 1 presents the average of the classification error, gap, eigengap and normalized cut over the ten samples using $\theta_{\alpha,CE}$ as the weights applied to the features. In this situation the $\theta_{\alpha,SA}$ are identical to the $\theta_{\alpha,CE}$. In all cases the learned parameters were positive and approximately equal for the meaningful features and 0 on the noisy dimensions, which lead to great clusterings on the test data samples.

Table 1: **Results for the bull's-eye experiments.** n_t is the size of the training data set the parameters were learned on and $Ndim$ is the number of noisy dimensions added to the data. The α presented here are those chosen based on the lowest training classification error. The classification error (CE), gap_θ, Δ_k and NCut are averaged over 10 independent test sets.

n_t	$Ndim$	α_{CE}	CE	gap_θ	Δ_k	Ncut
150	1	1.2	0	4.4e-5	1.1e-3	4.9e-3
200	2	1.2	0	9.3e-5	3.0e-4	7.0e-3
400	4	1.2	0	7.9e-5	5.4e-4	6.7e-3
700	16	2	0	1.1e-4	4.0e-4	7.2e-3

7.2 Letters

The second set of experiments is performed on the Letter Recognition data set from the UCI KDD archive [Slate, 1991]. Each letter has sixteen features, $y_{i,1:16}$, integer valued from 0 to 16, associated with it. Some examples of the attributes are the height and width of the box and the mean x and y positions of the "on"-pixels. The distance used for creating the $\mathbf{x}$ array is defined here:

$$x_{ij,f} = \begin{cases} 0 & \text{if } y_{i,f} = y_{i,f} = 0 \\ \frac{|y_{i,f} - y_{j,f}|}{y_{i,f} + y_{j,f}} & \text{otherwise} \end{cases}$$

This distance was chosen because it scales down the features into [0,1] and because it was believed that a small difference between two large attribute values was less informative than a small difference between two small attribute values. This data set complements the bull's eye

data set because the Multi-way Normalized cut is greater than 0, the clusters are dense and there are redundant features.

The experiments described here represent all of the experiments we have done on this data. Due to time and memory limitations small subsets of the letters were explored. Between the training and test sets each letter appears approximately 750 times. The training set of the 'SM' data consisted of 50 occurrences of each letter, the 'WA' and 'EI' training data contained 100 of each letter, and the training set for the 'ACI' and 'ACIM' data had 200 of each letter. We also chose to re-sample sets of 150 letters, 25 times from each of the test data set, to test the parameters on smaller data subsets, because it was noted on the artificial data that larger data sets provided for better clusterings even with the learned parameters applied.

Table 2 gives the results of the learning experiments for both $\theta_{\alpha,CE}$ and $\theta_{\alpha,SA}$ on the training and test data sets. The optimal α's chosen by the two methods differ, which is due in part to the fact that it is rare, in either method, to have a unique optimal α and ties were decided differently between the methods. A priori it might be thought that the α chosen from the CE on the training set might over fit the training data, but the test CE's do not show much evidence of over-fitting in these experiments. This maybe because the learning algorithm does not minimize the clustering error directly. To make sure that learned parameters was in fact improving the clustering, we clustered the re-sampled test data using parameter values all equal to 0.1, and these are presented in the last column of Table 2. Note also the consistent decrease in CE with the increase of the test size. We attribute this to the smoothing effect of a large sample size on the eigenvectors.

It is interesting to ask whether there is a common set of important features across letters. Figure 2 plots the parameters, $\theta_{\alpha,CE}$, assigned to each feature for the five different subsets tested. The $\theta_{\alpha,SA}$ are very similar and are not shown. While there is variation in the magnitude of the parameters, there seems to be some agreement in the weights selected as important. It appears that features twelve and fourteen are relatively important for all of the data sets, while one, two, five and nine do not appear to be very important for clustering these subsets. The parameters learned from the letters 'S' and 'M' appear to be the most different from the others. To further test this idea of a common set of features, the $\theta_{\alpha,SA}$ learned from the 'SM' subset, were applied to the other subsets and the clustering errors for the multiple disjoint test sets are reported in Table 2. They do not differ substantially suggesting that this data set has redundant features.

7.3 Stability Theorem

All of the experiments previously discussed chose clusterings by minimizing the gap, while maintaining a large

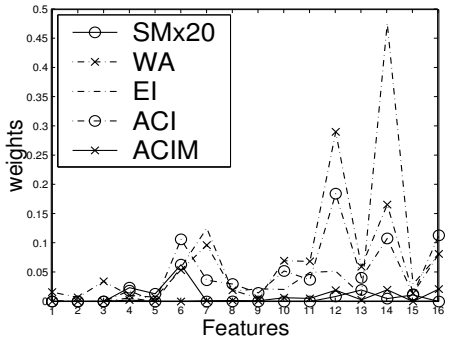


Figure 2: Comparison of learned parameters on all five data sets for the letters

eigengap. This set of experiments are designed to show that the clusterings achieved by optimizing this criterion are close to the optimal clustering. We performed a set of simple experiments, constructing a matrix S, resulting in a P with PCE's, to which we added random noise. Table 3 gives the value of the bound in Theorem 1 as a function of the noise, averaged over 10 noise realizations, for the clustering obtained by the Meilă-Shi spectral algorithm [Meilă and Shi, 2001b]; $K = 5$ for this data set. As one can see, the bound is informative up to significant noise levels (SNR of about 1). Since the node volumes are known, in the best cases the bound can represent a proof that the optimal clustering for this data set has been actually found.

8 Discussion and conclusions

We have introduced a new criterion for learning the similarity in spectral clustering. The criterion optimizes the quality of the target clustering, while constraining the parameters θ as little as possible in the process. This is achieved by choosing the gap as the clustering quality, and by adding the squared eigengap as a regularization term. One of the difficulties of learning in spectral clustering is the numerical optimization of the chosen criteria, as they often depend on θ through functions of the eigenvalues and vectors of P or another matrix. The gradients of such functions are expensive to compute and often unstable. Our choice of objective function also performs well in this respect, in that it can be optimized by a rather unsophisticated gradient descent algorithm. This is partly due to the eigengap term which has the effect of enhancing the numerical stability of the problem.

The amount of regularization is selected (semi-) automatically on the training set alone, with no further adjustments on the test set. Of course, some obvious variations are possible, like using multiple training sets, examining the δ_α, g_α graph on the test data, or other permissible tunings on the test data or on an independent validation set. We have avoided these here, as our focus was to validate the power of the regularization using training data

Table 2: **Results of Letter experiments**
The letter subsets are provided in the first column. The four columns titled $\theta_{\alpha,CE}$ present the α value chosen by the lowest training CE, the training and test classification errors (CE); and the "Resample" column gives the average CE and standard deviation in parenthesis over the 25 re-sampled data sets of 150 letters from the test data. The columns titled $\theta_{\alpha,SA}$ present these same results when the parameters corresponding to the α chosen by the Select Alpha method were applied. The column labeled "$\theta_{\alpha,SA}$ from SM" contains the average classification error over the re-sampled test sets when the parameters learned from the SM subset are applied to the other data sets. Finally, the last column presents the average CE when parameters all equal to 0.1 are applied to the re-sampled test data.

	$\theta_{\alpha,CE}$				$\theta_{\alpha,SA}$				$\theta_{\alpha,SA}$ from SM	$\theta_{1:16} = 0.1$
		Train CE	Test CE	Resample CE		Train CE	Test CE	Resample CE	Resample CE	Resample CE
	α	%	%	avg % (sd)	α	%	%	avg % (sd)	avg % (sd)	avg % (sd)
SM	4	0.0	2.8	6.8 (4.8)	2	11	2.8	5.7 (2.5)	NA	21.0 (12.9)
WA	10	2.0	3.2	4.6 (1.3)	0.8	4.0	5.3	5.8 (1.8)	4.8 (1.9)	8.8 (2.6)
ACI	4	8.3	6.6	8.1 (2.1)	8	8.5	7.8	12.0 (5.3)	9.7 (2.9)	21.3 (7.3)
AICM	8	13.4	12	17.3 (6.9)	1	16.3	14	15.2 (3.0)	32.3 (9.6)	32.2 (9.5)
EI	2	8.5	17.9	15.4 (9.2)	4	38	44	43.3 (7.1)	23.0 (2.5)	19.4 (4.0)

Table 3: Results of experiments showing that optimizing the criterion presented here, achieves clustering close to the optimal. The magnitude of the noise added to the S matrix is given in the column headings. The average bound (std. dev), defined in Theorem 1, over 10 replications at each noise level is presented in each column. We can see that up to noise magnitude of 3.2 the bound is informative.

Noise level	0.01	0.1	0.4	1.6	3.2	12.8	25.6
Bound (stdev)	1.7e-6 (1e-7)	3e-4 (3e-5)	0.004 (4e-4)	0.048 (0.006)	0.16 (0.028)	4.73 (6.8)	10.59 (6.1)

alone. The experimental results are very promising, as the algorithm clusters both sparse and blocky data, and eliminates the noisy features flawlessly.

The stability theorem 1 and its corollary can be used outside of spectral learning. For instance, the bound can tell one how far a given clustering is w.r.t the unknown optimal clustering on a data set, and even, in the luckiest cases, prove that the best clustering was found. The theorem makes no explicitly assumptions about the similarity matrix S. However, one should be aware that not every S will have a clustering good enough to satisfy the bound.

We conclude by remarking that from the perspective of learning, this work is just a beginning. A frame perhaps solid enough to allow one to think of the yet unanswered questions at the core of statistical learning, like sample complexity, prior knowledge, generalization bounds and so on. We hope that our future work will contribute to these areas.

Acknowledgments

The authors gratefully acknowledge Jim Burke for his invaluable advice on difficult optimizations. This work was partially supported by NSF VIGRE grant DMS-9810726, NSF ITR grant 0313339 and the University of Washington RRF grant 2684.

References

[Bach and Jordan, 2004] Bach, F. and Jordan, M. I. (2004). Learning spectral clustering. In Thrun, S. and Saul, L., editors, *Advances in Neural Information Processing Systems 16,*, Cambridge, MA. MIT Press.

[Bertsekas, 1999] Bertsekas, D. P. (1999). *Nonlinear programming.* Athena Scientific, Cambridge, MA, 2 edition.

[Fowlkes et al., 2004] Fowlkes, C., Belongie, S., Chung, F., and Malik, J. (2004). Spectral grouping using the nystrom method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225.

[Hubert and Arabie, 1985] Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2:193–218.

[Lancaster, 1969] Lancaster, H. (1969). *The Chi-Squared Distribution.* Wiley.

[Meilă, 2002] Meilă, M. (2002). The multicut lemma. Technical Report 417, University of Washington.

[Meilă and Shi, 2001a] Meilă, M. and Shi, J. (2001a). Learning segmentation by random walks. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems*, volume 13, pages 873–879, Cambridge, MA. MIT Press.

[Meilă and Shi, 2001b] Meilă, M. and Shi, J. (2001b). A random walks view of spectral segmentation. In Jaakkola, T. and Richardson, T., editors, *Artificial Intelligence and Statistics AISTATS.*

[Slate, 1991] Slate, D. (1991). UCI repository of machine learning databases.

[Verma and Meilă, 2003] Verma, D. and Meilă, M. (2003). A comparison of spectral clustering algorithms. TR 03-05-01, University of Washington. (sumitted).

[Yu and Shi, 2003] Yu, S. X. and Shi, J. (2003). Multiclass spectral clustering. In *International Conference on Computer Vision.*

Proof of Theorem 1

For any clustering $\mathcal{C}$ and fixed S matrix, with L defined as in section 5 we denote by e^k the indicator vector of cluster $C_k \in \mathcal{C}$, $y^k = D^{1/2}e^k$, $Y = [y^1 \ldots y^K]$, $\mu_k = 1 - \lambda_k(L)$, U = the orthonormal matrix formed with the eigenvectors of L as columns. Thus, $I - L = U\mathrm{diag}(\mu_1, \ldots \mu_n)U^T$. It is easy to show that $MNCut(\mathcal{C}) = \sum_{k=1}^{K} y^{kT}(I-L)y^k$ and $gap(\mathcal{C}) = MNCut(\mathcal{C}) - \sum_{k=1}^{K}\mu_k$. For two clusterings $\mathcal{C}$, $\mathcal{C}'$ we express their respective Y, Y' in the basis defined by U as $Y = UA$, $Y' = UA'$ with A, A' being $n \times K$ matrices of coefficients. Let

$$A = \begin{bmatrix} \tilde{A} \\ E \end{bmatrix} \quad A' = \begin{bmatrix} \tilde{A}' \\ E' \end{bmatrix} \tag{18}$$

with $\tilde{A}, \tilde{A}'$ $K \times K$ matrices.

Lemma 3 *If $gap(\mathcal{C}) < \varepsilon$, then $||E||_F^2 < \delta$, where $\delta = \varepsilon/\Delta_K$ and $||E||_F^2$ represents the Frobenius norm.*

Proof Denote by $A_{.k}$ the k-th column of A.

$$\sum_{k=1}^{K} y^{kT}(I-L)y^k = \sum_{k=1}^{K} A_{.k}^T U^T(I-L)UA_{.k} \tag{19}$$

$$= \sum_{k=1}^{K}\sum_{j=1}^{n} A_{jk}^2\mu_j \tag{20}$$

$$\geq \sum_{k=1}^{K}\sum_{j=1}^{K} A_{jk}^2\mu_j + \mu_{K+1}\underbrace{\sum_{k=1}^{K}\sum_{j=K+1}^{n} A_{jk}^2}_{||E||_F^2} \tag{21}$$

Now, using the hypothesis, we have

$$\sum_{k=1}^{K}\sum_{j=1}^{K} A_{jk}^2\mu_j + \mu_{K+1}||E||_F^2 \leq \sum_{k=1}^{K}\mu_k + \varepsilon$$

$$\mu_{K+1}||E||_F^2 \leq \sum_{k=1}^{K}\mu_k(1 - \sum_{j=1}^{K} A_{kj}^2) + \varepsilon \tag{22}$$

$$\leq \mu_K\left(K - \sum_{k=1}^{K}\sum_{j=1}^{K} A_{kj}^2\right) + \varepsilon \tag{23}$$

$$= \mu_K||E||_F^2 + \varepsilon \tag{24}$$

■

Lemma 4 *Assume that the conditions of theorem 1 hold and let $Y, A, \tilde{A}, E$ be as defined before. Let the SVD of $\tilde{A}$ be given by*

$$\tilde{A}^T\tilde{A} = \tilde{V}_1^T\mathrm{diag}\{\sigma_1^2, \sigma_2^2, \ldots \sigma_K^2\}\tilde{V}_1 \tag{25}$$

with $\tilde{V}_1$ a unitary matrix. Then $\sigma_k^2 > 1 - \delta$ for $k = 1, \ldots K$.

Proof The columns of A are orthonormal. Therefore

$$A^TA = I = \tilde{A}^T\tilde{A} + E^TE$$

or

$$\tilde{A}^T\tilde{A} = I - E^TE$$

Let e_k, $k = 1, \ldots K$ be the singular values of E. Then, there is a unitary matrix $\tilde{V}_2$ such that

$$\tilde{V}_2^T\tilde{A}^T\tilde{A}\tilde{V}_2 = I - \mathrm{diag}\{e_1^2, \ldots e_K^2\} \tag{26}$$

Since $||E||_F^2 < \delta$ we have that $\sum_{k=1}^{K} e_k^2 < \delta$ and therefore $e_k^2 < \delta$. From (26) we also have that $\sigma_k^2 = 1 - e_k^2$ and $\tilde{V}_2 = \tilde{V}_1$ which implies $\sigma_k^2 > 1 - \delta$ for all $k = 1, \ldots K$. ■

Note also that if $||E||_F^2, ||E'||_F^2 \leq \delta$, then by the Cauchy-Schwartz inequality $||E^TE'|| \leq \delta$.

Lemma 5 *Assume that the conditions of theorem 1 hold and let Y, A, $\tilde{A}$, E, Y', A', $\tilde{A}'$, E' and δ be as defined before. Then*

$$||Y^TY'||_F^2 \geq K - (\sqrt{K}+1)^2\delta \tag{27}$$

Proof Let

$$||Y^TY'||_F = ||A^TA'||_F \tag{28}$$

$$= ||\tilde{A}^T\tilde{A}' + E^TE'||_F \tag{29}$$

$$\geq \left|\,||\tilde{A}^T\tilde{A}'||_F - ||E^TE'||_F\,\right| \tag{30}$$

Let us look at the first term of the difference above.

$$||\tilde{A}^T\tilde{A}'||_F^2 = \sum_{i=1}^{K}\sum_{k=1}^{K}\left(\sum_{j=1}^{K} A_{ji}A'_{jk}\right)^2 = \sum_{k=1}^{K}||\tilde{A}^TA'_{.k}||_2^2 \tag{31}$$

By virtue of the singular value decomposition in (26) $\tilde{A}$ can be written as

$$\tilde{A} = \tilde{V}_3\mathrm{diag}\{\sigma_1, \sigma_2, \ldots \sigma_K\}\tilde{V}_4 \tag{32}$$

with $\tilde{V}_3, \tilde{V}_4$ complex unitary matrices. Therefore

$$||\tilde{A}^T\tilde{A}'_{.k}||_2^2 = ||\tilde{V}_4^T\mathrm{diag}\{\sigma_1, \sigma_2, \ldots \sigma_K\}\tilde{V}_3^T\tilde{A}'_{.k}||_2^2 \tag{33}$$

$$= ||\mathrm{diag}\{\sigma_1, \sigma_2, \ldots \sigma_K\}\tilde{V}_3^T\tilde{A}'_{.k}||_2^2 \tag{34}$$

$$\geq (1-\delta)||\tilde{V}_3^T\tilde{A}'_{.k}||_2^2 \tag{35}$$

$$= (1-\delta)||\tilde{A}'_{.k}||_2^2 \tag{36}$$

Then, using equation (31) and lemma 3 we obtain

$$||\tilde{A}^T\tilde{A}'||_F^2 \geq (1-\delta)\sum_{k=1}^{K}||\tilde{A}'_{.k}||_2^2 \tag{37}$$

$$\geq (1-\delta)(K-\delta) \tag{38}$$

Using now equation (30) above we obtain

$$||Y^TY'||_F^2 \geq \left(||\tilde{A}^T\tilde{A}'||_F - ||E^TE'||_F\right)^2 \tag{39}$$

$$\geq (\sqrt{(1-\delta)(K-\delta)} - \delta)^2 \tag{40}$$

$$\geq K - (\sqrt{K}+1)^2\delta \tag{41}$$

As $d = 1 - \frac{1}{K}||Y^TY'||_F^2$ and $(\sqrt{K}+1)^2/K < 3$ the proof of the theorem is finished.

Approximate Inference for Infinite Contingent Bayesian Networks

Brian Milch, Bhaskara Marthi, David Sontag, Stuart Russell, Daniel L. Ong and Andrey Kolobov
Computer Science Division
University of California
Berkeley, CA 94704
{milch,bhaskara,russell,dsontag,dlong,karaya1}@cs.berkeley.edu

Abstract

In many practical problems—from tracking aircraft based on radar data to building a bibliographic database based on citation lists—we want to reason about an unbounded number of unseen objects with unknown relations among them. Bayesian networks, which define a fixed dependency structure on a finite set of variables, are not the ideal representation language for this task. This paper introduces *contingent Bayesian networks* (CBNs), which represent uncertainty about dependencies by labeling each edge with a condition under which it is active. A CBN may contain cycles and have infinitely many variables. Nevertheless, we give general conditions under which such a CBN defines a unique joint distribution over its variables. We also present a likelihood weighting algorithm that performs approximate inference in finite time per sampling step on any CBN that satisfies these conditions.

1 Introduction

One of the central tasks an intelligent agent must perform is to make inferences about the real-world objects that underlie its observations. This type of reasoning has a wide range of practical applications, from tracking aircraft based on radar data to building a bibliographic database based on citation lists. To tackle these problems, it makes sense to use probabilistic models that represent uncertainty about the number of underlying objects, the relations among them, and the mapping from observations to objects.

Over the past decade, a number of probabilistic modeling formalisms have been developed that explicitly represent objects and relations. Most work has focused on scenarios where, for any given query, there is no uncertainty about the set of relevant objects. In extending this line of work to unknown sets of objects, we face a difficulty: unless we place an upper bound on the number of underlying objects, the resulting model has infinitely many variables. We have developed a formalism called BLOG (Bayesian LOGic) in which such infinite models can be defined concisely [7]. However, it is not obvious under what conditions such models define probability distributions, or how to do inference on them.

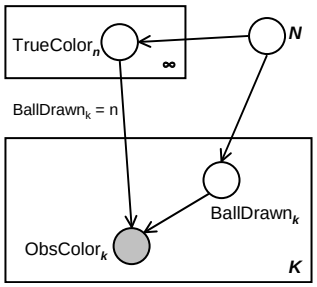


Figure 1: A graphical model (with plates representing repeated elements) for the balls-and-urn example. This is a BN if we disregard the labels $\mathsf{BallDrawn}_k = n$ on the edges $\mathsf{TrueColor}_n \rightarrow \mathsf{ObsColor}_k$ for $k \in \{1, \ldots, K\}$, $n \in \{1, 2, \ldots\}$. With the labels, it is a CBN.

Bayesian networks (BNs) with infinitely many variables are actually quite common: for instance, a dynamic BN with time running infinitely into the future has infinitely many nodes. These common models have the property that each node has only finitely many ancestors. So for finite sets of evidence and query variables, pruning away "barren" nodes [15] yields a finite BN that is sufficient for answering the query. However, generative probability models with unknown objects often involve infinite ancestor sets, as illustrated by the following stylized example from [13].

Example 1. *Suppose an urn contains some unknown number of balls N, and suppose our prior distribution for N assigns positive probability to every natural number. Each ball has a color—say, black or white—chosen independently from a fixed prior. Suppose we repeatedly draw a ball uniformly at random, observe its color, and return it to the urn. We cannot distinguish two identically colored balls from each other. Furthermore, we have some (known) probability of making a mistake in each color observation.*

Given our observations, we might want to predict the total number of balls in the urn, or solve the identity uncertainty *problem: computing the posterior probability that (for example) we drew the same ball on our first two draws.*

Fig. 1 shows a graphical model for this example. There is an infinite set of variables for the true colors of the balls; each $\mathsf{TrueColor}_n$ variable takes the special value null when $N < n$. Each $\mathsf{BallDrawn}_k$ variable takes a value between 1 and N, indicating the ball drawn on draw k. The $\mathsf{ObsColor}_k$ variable then depends on $\mathsf{TrueColor}_{(\mathsf{BallDrawn}_k)}$. In this BN, all the infinitely many $\mathsf{TrueColor}_n$ variables are ancestors of each $\mathsf{ObsColor}_k$ variable. Thus, even if we prune barren nodes, we cannot obtain a finite BN for computing the posterior over N. The same problem arises in real-world identity uncertainty tasks, such as resolving coreference among citations that refer to some underlying publications [10].

Bayesian networks also fall short in representing scenarios where the relations between objects or events—and thus the dependencies between random variables—are random.

Example 2. *Suppose a hurricane is going to strike two cities, Alphatown and Betaville, but it is not known which city will be hit first. The amount of damage in each city depends on the level of preparations made in each city. Also, the level of preparations in the second city depends on the amount of damage in the first city. Fig. 2 shows a model for this situation, where the variable F takes on the value A or B to indicate whether Alphatown or Betaville is hit first.*

In this example, suppose that we have a good estimate of the distribution for preparations in the first city, and of the conditional probability distribution (CPD) for preparations in the second city given damage in the first. The obvious graphical model to draw is the one in Fig. 2, but it has a figure-eight-shaped cycle. Of course, we can construct a BN for the intended distribution by choosing an arbitrary ordering of the variables and including all necessary edges to each variable from its predecessors. Suppose we use the ordering F, P_A, D_A, P_B, D_B. Then $P(P_A|F = A)$ is easy to write down, but to compute $P(P_A|F = B)$ we need to sum out P_B and D_B. There is no acyclic BN that reflects our causal intuitions.

Using a high-level modeling language, one can represent

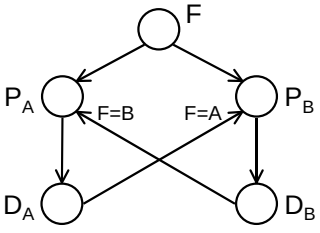


Figure 2: A cyclic BN for the hurricane scenario. P stands for preparations, D for damage, A for Alphatown, B for Betaville, and F for the city that is hit first.

scenarios such as those in Figs. 1 and 2 in a compact and natural way. However, as we have seen, the BNs corresponding to such models may contain cycles or infinite ancestor sets. The assumptions of finiteness and acyclicity are fundamental not just for BN inference algorithms, but also for the standard theorem that every BN defines a unique joint distribution.

Our approach to such models is based on the notion of context-specific independence (CSI) [1]. In the balls-and-urn example, in the context $\mathsf{BallDrawn}_k = n$, $\mathsf{ObsColor}_k$ has only one other ancestor — $\mathsf{TrueColor}_n$. Similarly, the BN in Fig. 2 is acyclic in the context $F = A$ and also in the context $F = B$. To exploit these CSI properties, we define two generalizations of BNs that make CSI explicit. The first is *partition-based models* (PBMs), where instead of specifying a set of parents for each variable, one specifies an arbitrary partition of the outcome space that determines the variable's CPD. In Sec. 2, we give an abstract criterion that guarantees that a PBM defines a unique joint distribution.

To prove more concrete results, we focus in Sec. 3 on the special case of *contingent Bayesian networks* (CBNs): possibly infinite BNs where some edges are labeled with conditions. CBNs combine the use of decision trees for CPDs [1] with the idea of labeling edges to indicate when they are active [3]. In Sec. 3, we provide general conditions under which a contingent BN defines a unique probability distribution, even in the presence of cycles or infinite ancestor sets. In Sec. 4 we explore the extent to which results about CBNs carry over to the more general PBMs. Then in Sec. 5 we present a sampling algorithm for approximate inference in contingent BNs. The time required to generate a sample using this algorithm depends only on the size of the context-specifically relevant network, not the total size of the CBN (which may be infinite). Experimental results for this algorithm are given in Sec. 6. We omit proofs for reasons of space; the proofs can be found in our technical report [8].

2 Partition-based models

We assume a set $\mathcal{V}$ of random variables, which may be countably infinite. Each variable X has a domain $\mathrm{dom}(X)$; we assume in this paper that each domain is at most countably infinite. The outcome space over which we would like to define a probability measure is the product space $\Omega \triangleq \times_{(X \in \mathcal{V})} \mathrm{dom}(X)$. An outcome $\omega \in \Omega$ is an assignment of values to all the variables; we write $X(\omega)$ for the value of X in ω.

An *instantiation* σ is an assignment of values to a subset of $\mathcal{V}$. We write $\mathrm{vars}(\sigma)$ for the set of variables to which σ assigns values, and σ_X for the value that σ assigns to a variable $X \in \mathrm{vars}(\sigma)$. The empty instantiation is denoted $\varnothing$. An instantiation σ is said to be finite if $\mathrm{vars}(\sigma)$ is finite. The *completions* of σ, denoted $\mathrm{comp}(\sigma)$, are those

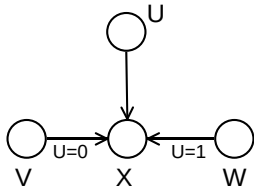


Figure 3: A simple contingent BN.

outcomes that agree with σ on $\mathrm{vars}(\sigma)$:

$$\mathrm{comp}(\sigma) \triangleq \{\omega \in \Omega : \forall X \in \mathrm{vars}(\sigma), X(\omega) = \sigma_X\}$$

If σ is a *full* instantiation — that is, $\mathrm{vars}(\sigma) = \mathcal{V}$ — then $\mathrm{comp}(\sigma)$ consists of just a single outcome.

To motivate our approach to defining a probability measure on Ω, consider the BN in Fig. 3, ignoring for now the labels on the edges. To completely specify this model, we would have to provide, in addition to the graph structure, a conditional probability distribution (CPD) for each variable. For example, assuming the variables are binary, the CPD for X would be a table with 8 rows, each corresponding to an instantiation of X's three parents. Another way of viewing this is that X's parent set defines a partition of Ω where each CPT row corresponds to a block (i.e., element) of the partition. This may seem like a pedantic rephrasing, but partitions can expose more structure in the CPD. For example, suppose X depends only on V when $U = 0$ and only on W when $U = 1$. The tabular CPD for X would still be the same size, but now the partition for X only has four blocks: $\mathrm{comp}(U = 0, V = 0)$, $\mathrm{comp}(U = 0, V = 1)$, $\mathrm{comp}(U = 1, W = 0)$, and $\mathrm{comp}(U = 1, W = 1)$.

Definition 1. *A* partition-based model Γ *over* $\mathcal{V}$ *consists of*

- *for each* $X \in \mathcal{V}$*, a partition* Λ_X^Γ *of* Ω *where we write* $\lambda_X^\Gamma(\omega)$ *to denote the block of the partition that the outcome* ω *belongs to;*
- *for each* $X \in \mathcal{V}$ *and block* $\lambda \in \Lambda_X^\Gamma$*, a probability distribution* $p_\Gamma(X|\lambda)$ *over* $\mathrm{dom}(X)$.

A PBM defines a probability distribution over Ω. If $\mathcal{V}$ is finite, this distribution can be specified as a product expression, just as for an ordinary BN:

$$P(\omega) \triangleq \prod_{X \in \mathcal{V}} p_\Gamma(X(\omega)|\lambda_X^\Gamma(\omega)) \quad (1)$$

Unfortunately, this equation becomes meaningless when $\mathcal{V}$ is infinite, because the probability of each outcome ω will typically be zero. A natural solution is to define the probabilities of finite instantiations, and then rely on Kolmogorov's extension theorem (see, e.g., [2]) to ensure that we have defined a unique distribution over outcomes. But Eq. 1 relies on having a full outcome ω to determine which CPD to use for each variable X.

How can we write a similar product expression that involves only a partial instantiation? We need the notion of a partial instantiation *supporting* a variable.

Definition 2. *In a PBM* Γ*, an instantiation* σ supports *a variable* X *if there is some block* $\lambda \in \Lambda_X^\Gamma$ *such that* $\mathrm{comp}(\sigma) \subseteq \lambda$*. In this case we write* $\lambda_X^\Gamma(\sigma)$ *for the unique element of* Λ_X^Γ *that has* $\mathrm{comp}(\sigma)$ *as a subset.*

Intuitively, σ supports X if knowing σ is enough to tell us which block of Λ_X^Γ we're in, and thus which CPD to use for X. In Fig. 3, $(U = 0, V = 0)$ supports X, but $(U = 1, V = 0)$ does not. In an ordinary BN, any instantiation of the parents of X supports X.

An instantiation σ is *self-supporting* if every $X \in \mathrm{vars}(\sigma)$ is supported by σ. In a BN, if $\mathbf{U}$ is an ancestral set (a set of variables that includes all the ancestors of its elements), then every instantiation of $\mathbf{U}$ is self-supporting.

Definition 3. *A probability measure* P *over* $\mathcal{V}$ satisfies *a PBM* Γ *if for every finite, self-supporting instantiation* σ*:*

$$P(\mathrm{comp}(\sigma)) = \prod_{X \in \mathrm{vars}(\ )} p_\Gamma(\sigma_X|\lambda_X^\Gamma(\sigma)) \quad (2)$$

A PBM is *well-defined* if there is a unique probability measure that satisfies it. One way a PBM can fail to be well-defined is if the constraints specified by Eq. 2 are inconsistent: for instance, if they require that the instantiations $(X = 1, Y = 1)$ and $(X = 0, Y = 0)$ both have probability 0.9. Conversely, a PBM can be satisfied by many distributions if, for example, the only self-supporting instantiations are infinite ones — then Def. 3 imposes no constraints.

When can we be sure that a PBM is well-defined? First, recall that a BN is well-defined if it is acyclic, or equivalently, if its nodes have a topological ordering. Thus, it seems reasonable to think about numbering the variables in a PBM. A *numbering* of $\mathcal{V}$ is a bijection π from $\mathcal{V}$ to some prefix of $\mathbb{N}$ (this will be a proper prefix if $\mathcal{V}$ is finite, and the whole set $\mathbb{N}$ if $\mathcal{V}$ is countably infinite). We define the predecessors of a variable X under π as:

$$\mathrm{Pred}_\pi[X] \triangleq \{U \in \mathcal{V} : \pi(U) < \pi(X)\}$$

Note that since each variable X is assigned a finite number $\pi(X)$, the predecessor set $\mathrm{Pred}_\pi[X]$ is always finite.

One of the purposes of PBMs is to handle cyclic scenarios such as Example 2. Thus, rather than speaking of a single topological numbering for a model, we speak of a *supportive numbering* for each outcome.

Definition 4. *A numbering* π *is a* supportive numbering *for an outcome* ω *if for each* $X \in \mathcal{V}$*, the instantiation* $\mathrm{Pred}_\pi[X](\omega)$ *supports* X.

Theorem 1. *A PBM* Γ *is well-defined if, for every outcome* $\omega \in \Omega$*, there exists a supportive numbering* π_ω.

The converse of this theorem is not true: a PBM may happen to be well-defined even if some outcomes do not have supportive numberings. But more importantly, the requirement that each outcome have a supportive numbering is very abstract. How could we determine whether it holds for a given PBM? To answer this question, we now turn to a more concrete type of model.

3 Contingent Bayesian networks

Contingent Bayesian networks (CBNs) are a special case of PBMs for which we can define more concrete well-definedness criteria, as well as an inference algorithm. In Fig. 3 the partition was represented not as a list of blocks, but implicitly by labeling each edge with an event. The meaning of an edge from W to X labeled with an event E, which we denote by $(W \to X \mid E)$, is that the value of W may be relevant to the CPD for X only when E occurs. In Fig. 3, W is relevant to X only when $U = 1$.

Using the definitions of $\mathcal{V}$ and Ω from the previous section, we can define a CBN structure as follows:

Definition 5. *A* CBN structure *$\mathcal{G}$ is a directed graph where the nodes are elements of $\mathcal{V}$ and each edge is labeled with a subset of Ω.*

In our diagrams, we leave an edge blank when it is labeled with the uninformative event Ω. An edge $(W \to X \mid E)$ is said to be *active* given an outcome ω if $\omega \in E$, and active given a partial instantiation σ if $\mathrm{comp}(\sigma) \subseteq E$. A variable W is an *active parent* of X given σ if an edge from W to X is active given σ.

Just as a BN is parameterized by specifying CPTs, a CBN is parameterized by specifying a decision tree for each node.

Definition 6. *A* decision tree *$\mathcal{T}$ is a directed tree where each node is an instantiation σ, such that:*

- *the root node is $\varnothing$;*
- *each non-leaf node σ* splits on *a variable $X^{\mathcal{T}}$ such that the children of σ are $\{(\sigma; X^{\mathcal{T}} = x) \ : \ x \in \mathrm{dom}\ X^{\mathcal{T}}\}$.*

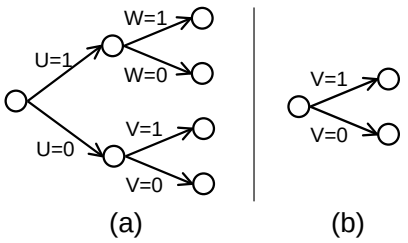


Figure 4: Two decision trees for X in Fig. 3. Tree (a) respects the CBN structure, while tree (b) does not.

Two decision trees are shown in Fig. 4. If a node splits on a variable that has infinitely many values, then it will have infinitely many children. This definition also allows a decision tree to contain infinite paths. However, each node in the tree is a finite instantiation, since it is connected to the root by a finite path. We will call a path *truncated* if it ends with a non-leaf node. Thus, a non-truncated path either continues infinitely or ends at a leaf. An outcome ω *matches* a path if ω is a completion of every node (instantiation) in the path. The non-truncated paths starting from the root are mutually exclusive and exhaustive, so a decision tree defines a partition of Ω.

Definition 7. *The partition $\Lambda_{\mathcal{T}}$ defined by a decision tree $\mathcal{T}$ consists of a block of the form $\{\omega \in \Omega \ : \ \omega$ matches $\}$ for each non-truncated path starting at the root of $\mathcal{T}$.*

So for each variable X, we specify a decision tree $\mathcal{T}_X$, thus defining a partition $\Lambda_X \triangleq \Lambda_{(\mathcal{T}_X)}$. To complete the parameterization, we also specify a function $p_{\mathcal{B}}(X = x|\lambda)$ that maps each $\lambda \in \Lambda_X$ to a distribution over $\mathrm{dom}\,(X)$. However, the decision tree for X must respect the CBN structure in the following sense.

Definition 8. *A decision tree $\mathcal{T}$* respects the CBN structure *$\mathcal{G}$ at X if for every node $\sigma \in \mathcal{T}$ that splits on a variable W, there is an edge $(W \to X \mid E)$ in $\mathcal{G}$ that is active given σ.*

For example, tree (a) in Fig. 4 respects the CBN structure of Fig. 3 at X. However, tree (b) does not: the root instantiation $\varnothing$ does not activate the edge $(V \to X \mid U = 0)$, so it should not split on V.

Definition 9. *A* contingent Bayesian network *(CBN) $\mathcal{B}$ over $\mathcal{V}$ consists of a CBN structure $\mathcal{G}^{\mathcal{B}}$, and for each variable $X \in \mathcal{V}$:*

- *a decision tree $\mathcal{T}_X^{\mathcal{B}}$ that respects $\mathcal{G}^{\mathcal{B}}$ at X, defining a partition $\Lambda_X^{\mathcal{B}} \triangleq \Lambda_{(\mathcal{T}_X^{\mathcal{B}})}$;*
- *for each block $\lambda \in \Lambda_X^{\mathcal{B}}$, a probability distribution $p_{\mathcal{B}}(X|\lambda)$ over $\mathrm{dom}\,(X)$.*

It is clear that a CBN is a kind of PBM, since it defines a partition and a conditional probability distribution for each variable. Thus, we can carry over the definitions from the previous section of what it means for a distribution to satisfy a CBN, and for a CBN to be well-defined.

We will now give a set of structural conditions that ensure that a CBN is well-defined. We call a set of edges in $\mathcal{G}$ *consistent* if the events on the edges have a non-empty intersection: that is, if there is some outcome that makes all the edges active.

Theorem 2. *Suppose a CBN $\mathcal{B}$ satisfies the following:*

(A1) No consistent path in $\mathcal{G}^{\mathcal{B}}$ forms a cycle.
(A2) No consistent path in $\mathcal{G}^{\mathcal{B}}$ forms an infinite receding chain $X_1 \quad X_2 \quad X_3 \quad \cdots$.
(A3) No variable $X \in \mathcal{V}$ has an infinite, consistent set of incoming edges in $\mathcal{G}^{\mathcal{B}}$.

Then $\mathcal{B}$ is well-defined.

A CBN that satisfies the conditions of Thm. 2 is said to be *structurally well-defined*. If a CBN has a finite set of variables, we can check the conditions directly. For instance, the CBN in Fig. 2 is structurally well-defined: although it contains a cycle, the cycle is not consistent.

The balls-and-urn example (Fig. 1) has infinitely many nodes, so we cannot write out the CBN explicitly. However, it is clear from the plates representation that this CBN is structurally well-defined as well: there are no cycles or infinite receding chains, and although each $\mathsf{ObsColor}_k$ node has infinitely many incoming edges, the labels $\mathsf{BallDrawn}_k = n$ ensure that exactly one of these edges is active in each outcome. In [8], we discuss the general problem of determining whether the infinite CBN defined by a high-level model is structurally well-defined.

4 CBNs as implementations of PBMs

In a PBM, we specify an arbitrary partition for each variable; in CBNs, we restrict ourselves to partitions generated by decision trees. But given any partition Λ, we can construct a decision tree $\mathcal{T}$ that yields a partition at least as fine as Λ—that is, such that each block $\lambda \in \Lambda_{\mathcal{T}}$ is a subset of some $\lambda' \in \Lambda$. In the worst case, every path starting at the root in $\mathcal{T}$ will need to split on every variable. Thus, every PBM is *implemented* by some CBN, in the following sense:

Definition 10. *A CBN $\mathcal{B}$* implements *a PBM Γ over the same set of variables $\mathcal{V}$ if, for each variable $X \in \mathcal{V}$, each block $\lambda \in \Lambda_X^{\mathcal{B}}$ is a subset of some block $\lambda' \in \Lambda_X^{\Gamma}$, and $p_{\mathcal{B}}(X|\lambda) = p_{\Gamma}(X|\lambda')$.*

Theorem 3. *If a CBN $\mathcal{B}$ implements a PBM Γ and $\mathcal{B}$ is structurally well-defined, then Γ is also well-defined, and $\mathcal{B}$ and Γ are satisfied by the same unique distribution.*

Thm. 3 gives us a way to show that a PBM Γ is well-defined: construct a CBN $\mathcal{B}$ that implements Γ, and then use Thm. 2 to show that $\mathcal{B}$ is structurally well-defined. However, the following example illustrates a complication:

Example 3. *Consider predicting who will go to a weekly book group meeting. Suppose it is usually Bob's responsibility to prepare questions for discussion, but if a historical fiction book is being discussed, then Alice prepares questions. In general, Alice and Bob each go to the meeting with probability 0.9. However, if the book is historical fiction and Alice isn't going, then the group will have no discussion questions, so the probability that Bob bothers to go is only 0.1. Similary, if the book is not historical fiction and Bob isn't going, then Alice's probability of going is 0.1. We will use H, G_A and G_B to represent the binary variables "historical fiction", "Alice goes", and "Bob goes".*

This scenario is most naturally represented by a PBM. The probability that Bob goes is 0.1 given $((H = 1) \wedge (G_A = 0))$ and 0.9 otherwise, so the partition for G_B has two blocks. The partition for G_A has two blocks as well.

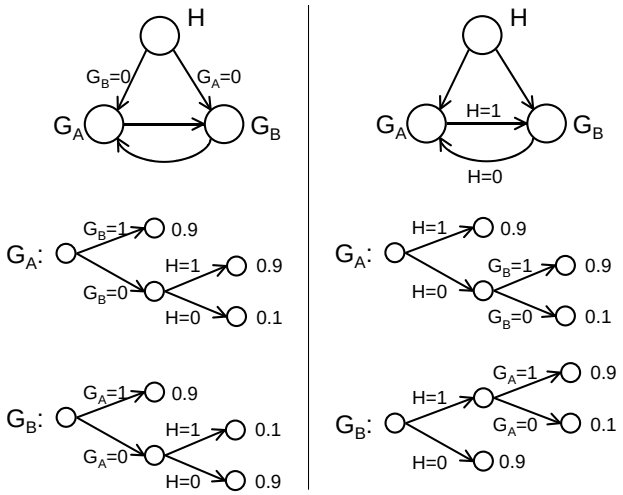


Figure 5: Two CBNs for Ex. 3, with decision trees and probabilities for G_A and G_B.

The CBNs in Fig. 5 both implement this PBM. There are no decision trees that yield exactly the desired partitions for G_A and G_B: the trees in Fig. 5 yield three blocks instead of two. Because the trees on the two sides of the figure split on the variables in different orders, they respect CBN structures with different labels on the edges. The CBN on the left has a consistent cycle, while the CBN on the right is structurally well-defined.

Thus, there may be multiple CBNs that implement a given PBM, and it may be that some of these CBNs are structurally well-defined while others are not. Even if we are given a well-defined PBM, it may be non-trivial to find a structurally well-defined CBN that implements it. Thus, algorithms that apply to structurally well-defined CBNs — such as the one we define in the next section — cannot be extended easily to general PBMs.

5 Inference

In this section we discuss an approximate inference algorithm for CBNs. To get information about a given CBN $\mathcal{B}$, our algorithm will use a few "black box" oracle functions. The function GET-ACTIVE-PARENT(X, σ) returns a variable that is an active parent of X given σ but is not already included in $\mathrm{vars}(\sigma)$. It does this by traversing the decision tree $\mathcal{T}_X^{\mathcal{B}}$, taking the branch associated with σ_U when the tree splits on a variable $U \in \mathrm{vars}(\sigma)$, until it reaches a split on a variable not included in $\mathrm{vars}(\sigma)$. If there is no such variable — which means that σ supports X — then it returns null. We also need the function COND-PROB(X, x, σ), which returns $p_{\mathcal{B}}(X = x|\sigma)$ whenever σ supports X, and the function SAMPLE-VALUE(X, σ), which randomly samples a value according to $p_{\mathcal{B}}(X|\sigma)$.

Our inference algorithm is a form of likelihood weight-

```
function CBN-LIKELIHOOD-WEIGHTING(Q, e, B, N)
  returns an estimate of P(Q|e)
  inputs: Q, the set of query variables
          e, evidence specified as an instantiation
          B, a contingent Bayesian network
          N, the number of samples to be generated

  W ← a map from dom(Q) to real numbers, with values
      lazily initialized to zero when accessed
  for j = 1 to N do
      , w ← CBN-WEIGHTED-SAMPLE(Q, e, B)
      W[q] ← W[q] + w where q =
  return NORMALIZE(W[Q])

function CBN-WEIGHTED-SAMPLE(Q, e, B)
  returns an instantiation and a weight

  ← ∅; stack ← an empty stack; w ← 1
  loop
    if stack is empty
      if some X in (Q ∪ vars(e)) is not in vars( )
        PUSH(X, stack)
      else
        return , w

    while X on top of stack is not supported by
        V ← GET-ACTIVE-PARENT(X, )
        push V on stack

    X ← POP(stack)
    if X in vars(e)
      x ← e_X
      w ← w × COND-PROB(X, x, )
    else
      x ← SAMPLE-VALUE(X, )
      ←( , X = x)
```

Figure 6: Likelihood weighting algorithm for CBNs.

ing. Recall that the likelihood weighting algorithm for BNs samples all non-evidence variables in topological order, then weights each sample by the conditional probability of the observed evidence [14]. Of course, we cannot sample all the variables in an infinite CBN. But even in a BN, it is not necessary to sample *all* the variables: the relevant variables can be found by following edges backwards from the query and evidence variables. We extend this notion to CBNs by only following edges that are active given the instantiation sampled so far. At each point in the algorithm (Fig. 6), we maintain an instantiation σ and a stack of variables that need to be sampled. If the variable X on the top of the stack is supported by σ, we pop X off the stack and sample it. Otherwise, we find a variable V that is an active parent of X given σ, and push V onto the stack. If the CBN is structurally admissible, this process terminates in finite time: condition (A1) ensures that we never push the same variable onto the stack twice, and conditions (A2) and (A3) ensure that the number of distinct variables pushed onto the stack is finite.

As an example, consider the balls-and-urn CBN (Fig. 1). If we want to query N given some color observations, the algorithm begins by pushing N onto the stack. Since N (which has no parents) is supported by $\varnothing$, it is immediately removed from the stack and sampled. Next, the first evidence variable $\mathsf{ObsColor}_1$ is pushed onto the stack. The active edge into $\mathsf{ObsColor}_1$ from $\mathsf{BallDrawn}_1$ is traversed, and $\mathsf{BallDrawn}_1$ is sampled immediately because it is supported by σ (which now includes N). The edge from $\mathsf{TrueColor}_n$ (for n equal to the sampled value of $\mathsf{BallDrawn}_1$) to $\mathsf{ObsColor}_1$ is now active, and so $\mathsf{TrueColor}_n$ is sampled as well. Now $\mathsf{ObsColor}_1$ is finally supported by σ, so it is removed from the stack and instantiated to its observed value. This process is repeated for all the observations. The resulting sample will get a high weight if the sampled true colors for the balls match the observed colors.

Intuitively, this algorithm is the same as likelihood weighting, in that we sample the variables in some topological order. The difference is that we sample only those variables that are needed to support the query and evidence variables, and we do not bother sampling any of the other variables in the CBN. Since the weight for a sample only depends on the conditional probabilities of the evidence variables, sampling additional variables would have no effect.

Theorem 4. *Given a structurally well-defined CBN $\mathcal{B}$, a finite evidence instantiation* **e***, a finite set* **Q** *of query variables, and a number of samples N, the algorithm* CBN-LIKELIHOOD-WEIGHTING *in Fig. 6 returns an estimate of the posterior distribution $P(\mathbf{Q}|\mathbf{e})$ that converges with probability 1 to the correct posterior as $N \to \infty$. Furthermore, each sampling step takes a finite amount of time.*

6 Experiments

We ran two sets of experiments using the likelihood weighting algorithm of Fig. 6. Both use the balls and urn setup from Ex. 1. The first experiment estimates the number of balls in the urn given the colors observed on 10 draws; the second experiment is an identity uncertainty problem. In both cases, we run experiments with both a noiseless sensor model, where the observed colors of balls always match their true colors, and a noisy sensor model, where with probability 0.2 the wrong color is reported.

The purpose of these experiments is to show that inference over an infinite number of variables can be done using a general algorithm in finite time. We show convergence of our results to the correct values, which were computed by enumerating equivalence classes of outcomes with up to 100 balls (see [8] for details). More efficient sampling algorithms for these problems have been designed by hand [9]; however, our algorithm is general-purpose, so it needs no modification to be applied to a different domain.

Number of balls: In the first experiment, we are predicting the total number of balls in the urn. The prior over the number of balls is a Poisson distribution with mean 6; each

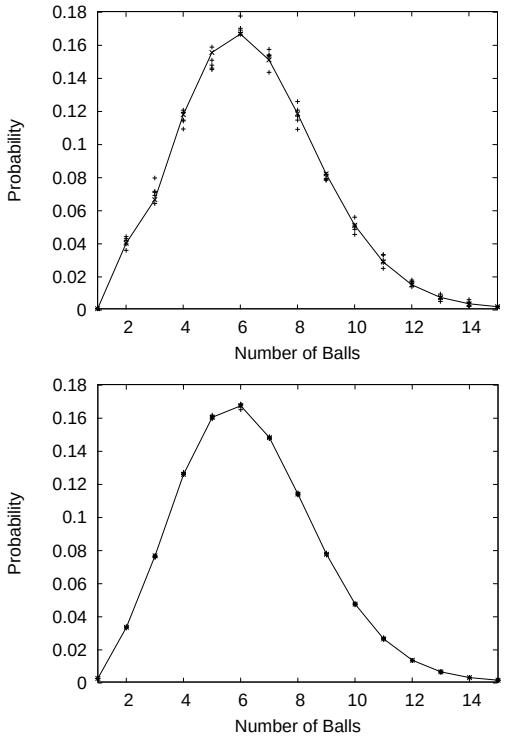


Figure 7: Posterior distributions for the total number of balls given 10 observations in the noise-free case (top) and noisy case (bottom). Exact probabilities are denoted by '$\times$'s and connected with a line; estimates from 5 sampling runs are marked with '+'s.

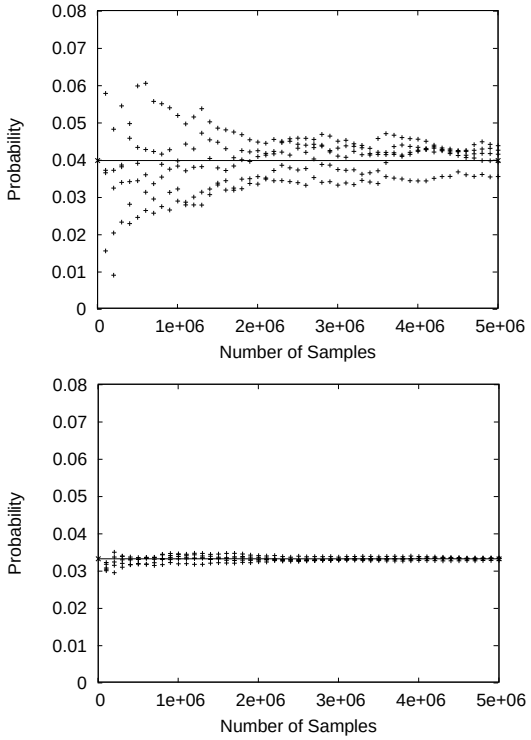


Figure 8: Probability that $N = 2$ given 10 observations (5 black, 5 white) in the noise-free case (top) and noisy case (bottom). Solid line indicates exact value; '+'s are values computed by 5 sampling runs at intervals of 100,000 samples.

ball is black with probability 0.5. The evidence consists of color observations for 10 draws from the urn: five are black and five are white. For each observation model, five independent trials were run, each of 5 million samples.[1]

Fig. 7 shows the posterior probabilities for total numbers of balls from 1 to 15 computed in each of the five trials, along with the exact probabilities. The results are all quite close to the true probability, especially in the noisy-observation case. The variance is higher for the noise-free model because the sampled true colors for the balls are often inconsistent with the observed colors, so many samples have zero weights.

Fig. 8 shows how quickly our algorithm converges to the correct value for a particular probability, $P(N = 2|\text{obs})$. The run with deterministic observations stays within 0.01 of the true probability after 2 million samples. The noisy-observation run converges faster, in just 100,000 samples.

Identity uncertainty: In the second experiment, three balls are drawn from the urn: a black one and then two white ones. We wish to find the probability that the second and third draws produced the same ball. The prior distribution over the number of balls is Poisson(6). Unlike the previous experiment, each ball is black with probability 0.3.

We ran five independent trials of 100,000 samples on the deterministic and noisy observation models. Fig. 9 shows the estimates from all five trials approaching the true probability as the number of samples increases. Note that again, the approximations for the noisy observation model converge more quickly. The noise-free case stays within 0.01 of the true probability after 70,000 samples, while the noisy case converges within 10,000 samples. Thus, we perform inference over a model with an unbounded number of objects and get reasonable approximations in finite time.

7 Related work

There are a number of formalisms for representing context-specific independence (CSI) in BNs. Boutilier et al. [1] use decision trees, just as we do in CBNs. Poole and Zhang [12] use a set of *parent contexts* (partial instantiations of the parents) for each node; such models can be represented as PBMs, although not necessarily as CBNs. Neither paper discusses infinite or cyclic models. The idea of labeling edges with the conditions under which they are active may have originated in [3] (a working paper that is no longer available); it was recently revived in [5].

[1] Our Java implementation averages about 1700 samples/sec. for the exact observation case and 1100 samples/sec. for the noisy observation model on a 3.2 GHz Intel Pentium 4.

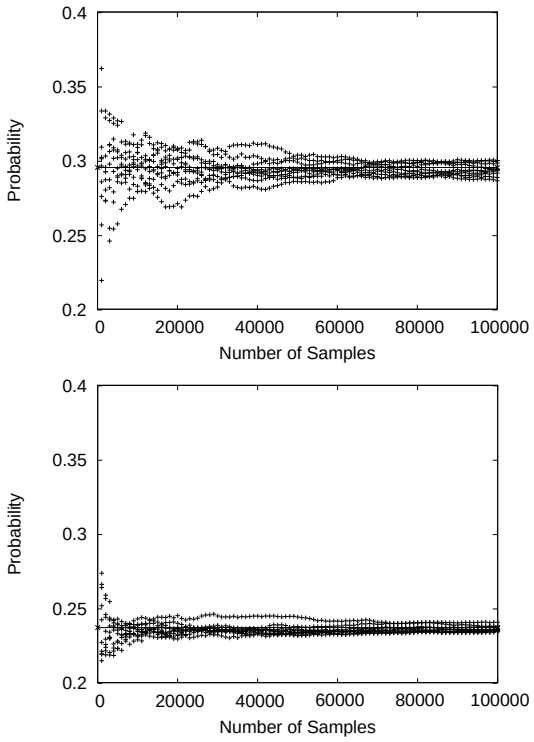


Figure 9: Probability that draws two and three produced the same ball for noise-free observations (top) and noisy observations (bottom). Solid line indicates exact value; '+'s are values computed by 5 sampling runs.

Bayesian multinets [4] can represent models that would be cyclic if they were drawn as ordinary BNs. A multinet is a mixture of BNs: to sample an outcome from a multinet, one first samples a value for the *hypothesis variable* H, and then samples the remaining variables using a hypothesis-specific BN. We could extend this approach to CBNs, representing a structurally well-defined CBN as a (possibly infinite) mixture of acyclic, finite-ancestor-set BNs. However, the number of hypothesis-specific BNs required would often be exponential in the number of variables that govern the dependency structure. On the other hand, to represent a given multinet as a CBN, we simply include an edge $V \rightarrow X$ with the label $H = h$ whenever that edge is present in the hypothesis-specific BN for h.

There has also been some work on handling infinite ancestor sets in BNs without representing CSI. Jaeger [6] states that an infinite BN defines a unique distribution if there is a well-founded topological ordering on its variables; that condition is more complete than ours in that it allows a node to have infinitely many active parents, but less complete in that it requires a single ordering for all contexts. Pfeffer and Koller [11] point out that a network containing an infinite receding path $X_1 \quad X_2 \quad X_3 \quad \cdots$ may still define a unique distribution if the CPDs along the path form a Markov chain with a unique stationary distribution.

8 Conclusion

We have presented contingent Bayesian networks, a formalism for defining probability distributions over possibly infinite sets of random variables in a way that makes context-specific independence explicit. We gave structural conditions under which a CBN is guaranteed to define a unique distribution—even if it contains cycles, or if some variables have infinite ancestor sets. We presented a sampling algorithm that is guaranteed to complete each sampling step in finite time and converge to the correct posterior distribution. We have also discussed how CBNs fit into the more general framework of partition-based models.

Our likelihood weighting algorithm, while completely general, is not efficient enough for most real-world problems. Our future work includes developing an efficient Metropolis-Hastings sampler that allows for user-specified proposal distributions; the results of [10] suggest that such a system can handle large inference problems satisfactorily. Further work at the theoretical level includes handling continuous variables, and deriving more complete conditions under which CBNs are guaranteed to be well-defined.

References

[1] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specifi c independence in Bayesian networks. In *Proc. 12th UAI*, pages 115–123, 1996.

[2] R. Durrett. *Probability: Theory and Examples*. Wadsworth, Belmont, CA, 2nd edition, 1996.

[3] R. M. Fung and R. D. Shachter. Contingent influence diagrams. Working Paper, Dept. of Engineering-Economic Systems, Stanford University, 1990.

[4] D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and Bayesian multinets. *AIJ*, 82(1–2):45–74, 1996.

[5] D. Heckerman, C. Meek, and D. Koller. Probabilistic models for relational data. Technical Report MSR-TR-2004-30, Microsoft Research, 2004.

[6] M. Jaeger. Reasoning about infi nite random structures with relational Bayesian networks. In *Proc. 6th KR*, 1998.

[7] B. Milch, B. Marthi, and S. Russell. BLOG: Relational modeling with unknown objects. In *ICML Wksp on Statistical Relational Learning*, 2004.

[8] B. Milch, B. Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Kolobov. BLOG: First-order probabilistic models with unknown objects. Technical report, UC Berkeley, 2004.

[9] H. Pasula. *Identity Uncertainty*. PhD thesis, UC Berkeley, 2003.

[10] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *NIPS 15*. MIT Press, Cambridge, MA, 2003.

[11] A. Pfeffer and D. Koller. Semantics and inference for recursive probability models. In *Proc. 17th AAAI*, 2000.

[12] D. Poole and N. L. Zhang. Exploiting contextual independence in probabilistic inference. *JAIR*, 18:263–313, 2003.

[13] S. Russell. Identity uncertainty. In *Proc. 9th Int'l Fuzzy Systems Assoc. World Congress*, 2001.

[14] S. Russell and P. Norvig. *Artifi cial Intelligence: A Modern Approach*. Morgan Kaufmann, 2nd edition, 2003.

[15] R. D. Shachter. Evaluating influence diagrams. *Op. Res.*, 34:871–882, 1986.

Hierarchical Probabilistic Neural Network Language Model

Frederic Morin
Dept. IRO, Université de Montréal
P.O. Box 6128, Succ. Centre-Ville,
Montreal, H3C 3J7, Qc, Canada
morinf@iro.umontreal.ca

Yoshua Bengio
Dept. IRO, Université de Montréal
P.O. Box 6128, Succ. Centre-Ville,
Montreal, H3C 3J7, Qc, Canada
Yoshua.Bengio@umontreal.ca

Abstract

In recent years, variants of a neural network architecture for statistical language modeling have been proposed and successfully applied, e.g. in the language modeling component of speech recognizers. The main advantage of these architectures is that they learn an embedding for words (or other symbols) in a continuous space that helps to smooth the language model and provide good generalization even when the number of training examples is insufficient. However, these models are extremely slow in comparison to the more commonly used n-gram models, both for training and recognition. As an alternative to an importance sampling method proposed to speed-up training, we introduce a hierarchical decomposition of the conditional probabilities that yields a speed-up of about 200 both during training and recognition. The hierarchical decomposition is a binary hierarchical clustering constrained by the prior knowledge extracted from the WordNet semantic hierarchy.

1 INTRODUCTION

The curse of dimensionality hits hard statistical language models because the number of possible combinations of n words from a dictionnary (e.g. of 50,000 words) is immensely larger than all the text potentially available, at least for $n > 2$. The problem comes down to transfering probability mass from the tiny fraction of observed cases to all the other combinations. From the point of view of machine learning, it is interesting to consider the different principles at work in obtaining such generalization. The most fundamental principle, used explicitly in non-parametric models, is that of similarity: if two objects are similar they should have a similar probability. Unfortunately, using a knowledge-free notion of similarity does not work well in high-dimensional spaces such as sequences of words. In the case of statistical language models, the most successful generalization principle (and corresponding notion of similarity) is also a very simple one, and it is used in interpolated and back-off n-gram models (Jelinek and Mercer, 1980; Katz, 1987): sequences that share shorter subsequences are similar and should share probability mass.

However, these methods are based on exact matching of subsequences, whereas it is obvious that two word sequences may not match and yet be very close to each other semantically. Taking this into account, another principle that has been shown to be very successful (in combination with the first one) is based on a notion of similarity between individual words: two word sequences are said to be "similar" if their corresponding words are "similar". Similarity between words is usually defined using **word classes** (Brown et al., 1992; Goodman, 2001b). These word classes correspond to a partition of the set of words in such a way that words in the same class share statistical properties in the context of their use, and this partition can be obtained with various clustering algorithms. This is a discrete all-or-nothing notion of similarity. Another way to define similarity between words is based on assigning each word to a continuous-valued set of features, and comparing words based on this feature vector. This idea has already been exploited in information retrieval (Schutze, 1993; Deerwester et al., 1990) using a singular value decomposition of a matrix of occurrences, indexed by words in one dimension and by documents in the other.

This idea has also been exploited in (Bengio, Ducharme and Vincent, 2001; Bengio et al., 2003): a neural network architecture is defined in which the first layer maps word symbols to their continuous representation as feature vectors, and the rest of the neural network is conventional and used to construct conditional probabilities of the next word given the previous ones. This model is described in detail in Section 2. The idea is to exploit the smoothness of the neural network to make sure that sequences of words that are similar according to this learned metric will be assigned a similar probability. Note that both the feature vec-

tors and the part of the model that computes probabilities from them are estimated jointly, by regularized maximum likelihood. This type of model is also related to the popular maximum entropy models (Berger, Della Pietra and Della Pietra, 1996) since the latter correspond to a neural network with no hidden units (the unnormalized log-probabilities are linear functions of the input indicators of presence of words).

This neural network approach has been shown to generalize well in comparison to interpolated n-gram models and class-based n-grams (Brown et al., 1992; Pereira, Tishby and Lee, 1993; Ney and Kneser, 1993; Niesler, Whittaker and Woodland, 1998; Baker and McCallum, 1998), both in terms of perplexity and in terms of classification error when used in a speech recognition system (Schwenk and Gauvain, 2002; Schwenk, 2004; Xu, Emami and Jelinek, 2003). In (Schwenk and Gauvain, 2002; Schwenk, 2004), it is shown how the model can be used to directly improve speech recognition performance. In (Xu, Emami and Jelinek, 2003), the approach is generalized to form the various conditional probability functions required in a stochastic parsing model called the Structured Language Model, and experiments also show that speech recognition performance can be improved over state-of-the-art alternatives. However, a major weakness of this approach is the very long training time as well as the large amount of computations required to compute probabilities, e.g. at the time of doing speech recognition (or any other application of the model). Note that such models could be used in other applications of statistical language modeling, such as automatic translation and information retrieval, but improving speed is important to make such applications possible.

The objective of this paper is thus to propose a much faster variant of the neural probabilistic language model. It is based on an idea that could in principle deliver close to exponential speed-up with respect to the number of words in the vocabulary. Indeed the computations required during training and during probability prediction are a small constant plus a factor linearly proportional to the number of words $|V|$ in the vocabulary V. The approach proposed here can yield a speed-up of order $O(\frac{|V|}{\log |V|})$ for the second term. It follows up on a proposal made in (Goodman, 2001b) to rewrite a probability function based on a partition of the set of words. The basic idea is to form a hierarchical description of a word as a sequence of $O(\log |V|)$ decisions, and to learn to take these probabilistic decisions instead of directly predicting each word's probability. Another important idea of this paper is to reuse the same model (i.e. the same parameters) for all those decisions (otherwise a very large number of models would be required and the whole model would not fit in computer memory), using a special symbolic input that characterizes the nodes in the tree of the hierarchical decomposition. Finally, we use prior knowledge in the WordNet lexical reference system to help define the hierarchy of word classes.

2 PROBABILISTIC NEURAL LANGUAGE MODEL

The objective is to estimate the joint probability of sequences of words and we do it through the estimation of the conditional probability of the next word (the **target word**) given a few previous words (the **context**):

$$P(w_1, \ldots, w_l) = \prod_t P(w_t | w_{t-1}, \ldots, w_{t-n+1}),$$

where w_t is the word at position t in a text and $w_t \in V$, the vocabulary. The conditional probability is estimated by a normalized function $f(w_t, w_{t-1}, \ldots, w_{t-n+1})$, with $\sum_v f(v, w_{t-1}, \ldots, w_{t-n+1}) = 1$.

In (Bengio, Ducharme and Vincent, 2001; Bengio et al., 2003) this conditional probability function is represented by a neural network with a particular structure. Its most important characteristic is that each input of this function (a word symbol) is first embedded into a Euclidean space (by learning to associate a real-valued "feature vector" to each word). The set of feature vectors for all the words in V is part of the set of parameters of the model, estimated by maximizing the empirical log-likelihood (minus weight decay regularization). The idea of associating each symbol with a distributed continuous representation is not new and was advocated since the early days of neural networks (Hinton, 1986; Elman, 1990). The idea of using neural networks for language modeling is not new either (Miikkulainen and Dyer, 1991; Xu and Rudnicky, 2000), and is similar to proposals of character-based text compression using neural networks to predict the probability of the next character (Schmidhuber, 1996).

There are two variants of the model in (Bengio, Ducharme and Vincent, 2001; Bengio et al., 2003): one with $|V|$ outputs with softmax normalization (and the target word w_t is not mapped to a feature vector, only the context words), and one with a single output which represents the unnormalized probability for w_t given the context words. Both variants gave similar performance in the experiments reported in (Bengio, Ducharme and Vincent, 2001; Bengio et al., 2003). We will start from the second variant here, which can be formalized as follows, using the Boltzmann distribution form, following (Hinton, 2000):

$$f(w_t, w_{t-1}, \ldots, w_{t-n+1}) = \frac{e^{-g(w_t, w_{t-1}, \ldots, w_{t-n+1})}}{\sum_v e^{-g(v, w_{t-1}, \ldots, w_{t-n+1})}}$$

where $g(v, w_{t-1}, \ldots, w_{t-n+1})$ is a learned function that can be interpreted as an energy, which is low when the tuple $(v, w_{t-1}, \ldots, w_{t-n+1})$ is "plausible".

Let F be an embedding matrix (a parameter) with row F_i the embedding (feature vector) for word i. The above energy function is represented by a first transformation of the input label through the feature vectors F_i, followed by an ordinary feedforward neural network (with a single output and a bias dependent on v):

$$g(v, w_{t-1}, \ldots, w_{t-n+1}) = a'. \tanh(c + Wx + UF'_v) + b_v \quad (1)$$

where x' denotes the transpose of x, $\tanh$ is applied element by element, a, c and b are parameters vectors, W and U are weight matrices (also parameters), and x denotes the concatenation of input feature vectors for context words:

$$x = (F_{w_{t-1}}, \ldots, F_{w_{t-n+1}})'. \quad (2)$$

Let h be the number of hidden units (the number of rows of W) and d the dimension of the embedding (number of columns of F). Computing $f(w_t, w_{t-1}, \ldots, w_{t-n+1})$ can be done in two steps: first compute $c + Wx$ (requires $hd(n-1)$ multiply-add operations), and second, for each $v \in V$, compute UF'_v (hd multiply-add operations) and the value of $g(v, ...)$ (h multiply-add operations). Hence total computation time for computing f is on the order of $(n-1)hd + |V|h(d+1)$. In the experiments reported in (Bengio et al., 2003), n is around 5, $|V|$ is around 20000, h is around 100, and d is around 30. This gives around 12000 operations for the first part (independent of $|V|$) and around 60 million operations for the second part (that is linear in $|V|$).

Our *goal in this paper is to drastically reduce the second part*, ideally by replacing the $O(|V|)$ computations by $O(\log |V|)$ computations.

3 HIERARCHICAL DECOMPOSITION CAN PROVIDE EXPONENTIAL SPEED-UP

In (Goodman, 2001b) it is shown how to speed-up a maximum entropy class-based statistical language model by using the following idea. Instead of computing directly $P(Y|X)$ (which involves normalization across all the values that Y can take), one defines a clustering partition for the Y (into the word classes C, such that there is a deterministic function $c(.)$ mapping Y to C), so as to write

$$P(Y = y|X = x) = P(Y = y|C = c(y), X)P(C = c(y)|X = x).$$

This is always true for any function $c(.)$ because $P(Y|X) = \sum_i P(Y, C = i|X) = \sum_i P(Y|C = i, X)P(C = i|X) = P(Y|C = c(Y), X)P(C = c(Y)|X)$ because only one value of C is compatible with the value of Y, the value $C = c(Y)$.

Although any $c(.)$ would yield correct probabilities, generalization could be better for choices of word classes that "make sense", i.e. those for which it easier to learn the $P(C = c(y)|X = x)$.

If Y can take 10000 values and we have 100 classes with 100 words y in each class, then instead of doing normalization over 10000 choices we only need to do two normalizations, each over 100 choices. If computation of conditional probabilities is proportional to the number of choices then the above would reduce computation by a factor 50. This is approximately what is gained according to the measurements reported in (Goodman, 2001b). The same paper suggests that one could introduce more levels to the decomposition and here we push this idea to the limit. Indeed, whereas a one-level decomposition should provide a speed-up on the order of $\frac{|V|}{\sqrt{|V|}} = \sqrt{|V|}$, a hierarchical decomposition represented by a balanced binary tree should provide an exponential speed-up, on the order of $\frac{|V|}{\log_2 |V|}$ (at least for the part of the computation that is linear in the number of choices).

Each word v must be represented by a bit vector $(b_1(v), \ldots b_m(v))$ (where m depends on v). This can be achieved by building a binary hierarchical clustering of words, and a method for doing so is presented in the next section. For example, $b_1(v) = 1$ indicates that v belongs to the top-level group 1 and $b_2(v) = 0$ indicates that it belongs to the sub-group 0 of that top-level group.

The next-word conditional probability can thus be represented and computed as follows:

$$P(v|w_{t-1}, \ldots, w_{t-n+1}) = \prod_{j=1}^{m} P(b_j(v)|b_1(v), \ldots, b_{j-1}(v), w_{t-1}, \ldots, w_{t-n+1})$$

This can be interpreted as a series of binary stochastic decisions associated with nodes of a binary tree. Each node is indexed by a bit vector corresponding to the path from the root to the node (append 1 or 0 according to whether the left or right branch of a decision node is followed). Each leaf corresponds to a word. If the tree is balanced then the maximum length of the bit vector is $\lceil \log_2 |V| \rceil$. Note that we could further reduce computation by looking for an encoding that takes the frequency of words into account, to reduce the average bit length to the unconditional entropy of words. For example with the corpus used in our experiments, $|V| = 10000$ so $\log_2 |V| \approx 13.3$ while the unigram entropy is about 9.16, i.e. a possible additional speed-up of 31% when taking word frequencies into account to better balance the binary tree. The gain would be greater for

larger vocabularies, but not a very significant improvement over the major one obtained by using a simple balanced hierarchy.

The "target class" (0 or 1) for each node is obtained directly from the target word in each context, using the bit encoding of that word. Note also that there will be a target (and gradient propagation) only for the nodes on the path from the root to the leaf associated with the target word. This is the major source of savings during training.

During recognition and testing, there are two main cases to consider: one needs the probability of only one word, e.g. the observed word, (or very few) , or one needs the probabilities of all the words. In the first case (which occurs during testing on a corpus) we still obtain the exponential speed-up. In the second case, we are back to $O(|V|)$ computations (with a constant factor overhead). For the purpose of estimating generalization performance (out-of-sample log-likelihood) only the probability of the observed next word is needed. And in practical applications such as speech recognition, we are only interested in discriminating between a few alternatives, e.g. those that are consistent with the acoustics, and represented in a treillis of possible word sequences.

This speed-up should be contrasted with the one provided by the importance sampling method proposed in (Bengio and Senécal, 2003). The latter method is based on the observation that the log-likelihood gradient is the average over the model's distribution for $P(v|context)$ of the energy gradient associated with all the possible next-words v. The idea is to approximate this average by a biased (but asymptotically unbiased) importance sampling scheme. This approach can lead to significant speed-up during training, but because the architecture is unchanged, probability computation during recognition and test still requires $O(|V|)$ computations for each prediction. Instead, the architecture proposed here gives significant speed-up both during training and test / recognition.

4 SHARING PARAMETERS ACROSS THE HIERARCHY

If a separate predictor is used for each of the nodes in the hierarchy, about $2|V|$ predictors will be needed. This represents a huge capacity since each predictor maps from the context words to a single probability. This might create problems in terms of computer memory (not all the models would fit at the same time in memory) as well as overfitting. Therefore we have chosen to build a model in which parameters are shared across the hierarchy. There are clearly many ways to achieve such sharing, and alternatives to the architecture presented here should motivate further study.

Based on our discussion in the introduction, it makes sense to force the word embedding to be shared across all nodes. This is important also because the matrix of word features F is the largest component of the parameter set.

Since each node in the hierarchy presumably has a semantic meaning (being associated with a group of hopefully similar-meaning words) it makes sense to also associate each node with a feature vector. Without loss of generality, we can consider the model to predict $P(b|node, w_{t-1}, \ldots, w_{t-n+1})$ where $node$ corresponds to a sequence of bits specifying a node in the hierarchy and b is the next bit (0 or 1), corresponding to one of the two children of $node$. This can be represented by a model similar to the one described in Section 2 and (Bengio, Ducharme and Vincent, 2001; Bengio et al., 2003) but with two kinds of symbols in input: the context words and the current node. We allow the embedding parameters for word cluster nodes to be different from those for words. Otherwise the architecture is the same, with the difference that there are only two choices to predict, instead of $|V|$ choices.

More precisely, the specific predictor used in our experiments is the following:

$$P(b = 1|node, w_{t-1}, \ldots, w_{t-n+1}) = \\ \mathrm{sigmoid}(\alpha_{node} + \beta'.\tanh(c + Wx + UN_{node}))$$

where x is the concatenation of context word features as in eq. 2, $\mathrm{sigmoid}(y) = 1/(1 + exp(-y))$, α_i is a bias parameter playing the same role as b_v in eq. 1, β is a weight vector playing the same role as a in eq. 1, c, W, U and F play the same role as in eq. 1, and N gives feature vector embeddings for nodes in a way similar that F gave feature vector embeddings for next-words in eq. 1.

5 USING WORDNET TO BUILD THE HIERARCHICAL DECOMPOSITION

A very important component of the whole model is the choice of the words binary encoding, i.e. of the hierarchical word clustering. In this paper we combine empirical statistics with prior knowledge from the WordNet resource (Fellbaum, 1998). Another option would have been to use a purely data-driven hierarchical clustering of words, and there are many other ways in which the WordNet resource could have been used to influence the resulting clustering.

The IS-A taxonomy in WordNet organizes semantic concepts associated with senses in a graph that is almost a tree. For our purposes we need a tree, so we have manually selected a parent for each of the few nodes that have more than one parent. The leaves of the WordNet taxonomy are senses and each word can be associated with more than one

sense. Words sharing the same sense are considered to be synonymous (at least in one of their uses). For our purpose we have to choose one of the senses for each word (to make the whole hierarchy one over words) and we selected the most frequent sense. A straightforward extension of the proposed model would keep the semantic ambiguity of each word: each word would be associated with several leaves (senses) of the WordNet hierarchy. This would require summing over all those leaves (and corresponding paths to the root) when computing next-word probabilities.

Note that the WordNet tree is not binary: each node may have many more than two children (this is particularly a problem for verbs and adjectives, for which WordNet is shallow and incomplete). To transform this hierarchy into a binary tree we perform a data-driven binary hierarchical clustering of the children associated with each node, as illustrated in Figure 1. The K-means algorithm is used at each step to split each cluster. To compare nodes, we associate each node with the subset of words that it covers. Each word is associated with a TF/IDF (Salton and Buckley, 1988) vector of document/word occurrence counts, where each "document" is a paragraph in the training corpus. Each node is associated with the dimension-wise median of the TF/IDF scores. Each TF/IDF score is the occurrence frequency of the word in the document times the logarithm of the ratio of the total number of documents by the number of documents containing the word.

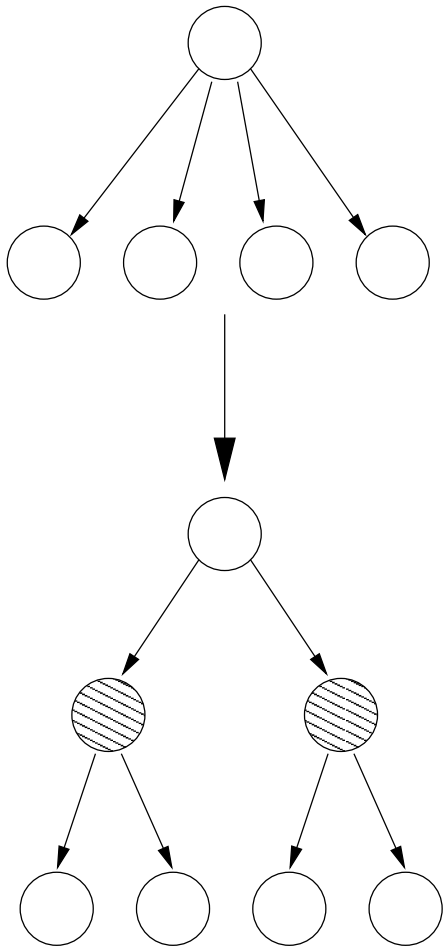


Figure 1: *WordNet's IS-A hierarchy is not a binary tree: most nodes have many children. Binary hierarchical clustering of these children is performed.*

6 COMPARATIVE RESULTS

Experiments were performed to evaluate the speed-up and any change in generalization error. The experiments also compared an alternative speed-up technique (Bengio and Senécal, 2003) that is based on importance sampling (but only provides a speed-up during training). The experiments were performed on the Brown corpus, with a reduced vocabulary size of 10,000 words (the most frequent ones). The corpus has 1,105,515 occurrences of words, split into 3 sets: 900,000 for training, 100,000 for validation (model selection), and 105,515 for testing. The validation set was used to select among a small number of choices for the size of the embeddings and the number of hidden units.

The results in terms of raw computations (time to process one example), either during training or during test are shown respectively in Tables 1 and 2. The computations were performed on Athlon processors with a 1.2 GHz clock. The speed-up during training is by a factor greater than 250 and during test by a factor close to 200. These are impressive but less than the $|V|/\log_2|V| \approx 750$ that could be expected if there was no overhead and no constant term in the computational cost.

It is also important to verify that learning still works and that the model generalizes well. As usual in statistical language modeling this is measured by the model's *perplexity* on the test data, which is the exponential of the average negative log-likehood on that data set. Training is performed over about 20 to 30 epochs according to validation set perplexity (early stopping). Table 3 shows the comparative generalization performance of the different architectures, along with that of an interpolated trigram and a class-based n-gram (same procedures as in (Bengio et al., 2003), which follow respectively (Jelinek and Mercer, 1980) and (Brown et al., 1992; Ney and Kneser, 1993; Niesler, Whittaker and Woodland, 1998)). The validation set was used to choose the order of the n-gram and the number of word classes for the class-based models. We used the implementation of these algorithms in the SRI Language Modeling toolkit, described by (Stolcke, 2002) and in `www.speech.sri.com/projects/srilm/`. Note that better performance should be obtainable with some of the tricks in (Goodman, 2001a). Combining the neural network with a trigram should also decrease its per-

architecture	Time per epoch (s)	Time per ex. (ms)	speed-up
original neural net	416 300	462.6	1
importance sampling	6 062	6.73	68.7
hierarchical model	1 609	1.79	258

Table 1: *Training time per epoch (going once through all the training examples) and per example. The original neural net is as described in sec. 2. The importance sampling algorithm (Bengio and Senécal, 2003) trains the same model faster. The hierarchical model is the one proposed here, and it yields a speed-up not only during training but for probability predictions as well (see the next table).*

architecture	Time per example (ms)	speed-up
original neural net	270.7	1
importance sampling	221.3	1.22
hierarchical model	1.4	193

Table 2: *Test time per example for the different algorithms. See Table 1's caption. It is at test time that the hierarchical model's advantage becomes clear in comparison to the importance sampling technique, since the latter only brings a speed-up during training.*

plexity, as already shown in (Bengio et al., 2003).

As shown in Table 3, the hierarchical model does not generalize as well as the original neural network, but the difference is not very large and still represents an improvement over the benchmark n-gram models. Given the very large speed-up, it is certainly worth investigating variations of the hierarchical model proposed here (in particular how to define the hierarchy) for which generalization could be better. Note also that the speed-up would be greater for larger vocabularies (e.g. 50,000 is not uncommon in speech recognition systems).

7 CONCLUSION AND FUTURE WORK

This paper proposes a novel architecture for speeding-up neural networks with a huge number of output classes and shows its usefulness in the context of statistical language modeling (which is a component of speech recognition and automatic translation systems). This work pushes to the limit a suggestion of (Goodman, 2001b) but also introduces the idea of sharing the same model for all nodes of the decomposition, which is more practical when the number of nodes is very large (tens of thousands here). The implementation and the experiments show that a very significant speed-up of around 200-fold can be achieved, with only a little degradation in generalization performance.

	Validation perplexity	Test perplexity
trigram	299.4	268.7
class-based	276.4	249.1
original neural net	213.2	195.3
importance sampling	209.4	192.6
hierarchical model	241.6	220.7

Table 3: *Test perplexity for the different architectures and for an interpolated trigram. The hierarchical model performed a bit worse than the original neural network, but is still better than the baseline interpolated trigram and the class-based model.*

From a linguistic point of view, one of the weaknesses of the above model is that it considers word clusters as deterministic functions of the word, but uses the nodes in WordNet's taxonomy to help define those clusters. However, WordNet provides word sense ambiguity information which could be used for linguistically more accurate modeling. The hierarchy would be a sense hierarchy instead of a word hiearchy, and each word would be associated with a number of senses (those allowed for that word in WordNet). In computing probabilities, this would involve summing over several paths from the root, corresponding to the different possible senses of the word. As a side effect, this could provide a word sense disambiguation model, and it could be trained both on sense-tagged supervised data and on unlabeled ordinary text. Since the average number of senses per word is small (less than a handful), the loss in speed would correspondingly be small.

Acknowledgments

The authors would like to thank the following funding organizations for support: NSERC, MITACS, IRIS, and the Canada Research Chairs.

References

Baker, D. and McCallum, A. (1998). Distributional clustering of words for text classification. In *SIGIR'98*.

Bengio, Y., Ducharme, R., and Vincent, P. (2001). A neural probabilistic language model. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 933–938. MIT Press.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Bengio, Y. and Senécal, J.-S. (2003). Quick training of probabilistic neural nets by importance sampling. In

Proceedings of AISTATS'2003.

Berger, A., Della Pietra, S., and Della Pietra, V. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22:39–71.

Brown, P., Pietra, V. D., DeSouza, P., Lai, J., and Mercer, R. (1992). Class-based *n*-gram models of natural language. *Computational Linguistics*, 18:467–479.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Elman, J. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. MIT Press.

Goodman, J. (2001a). A bit of progress in language modeling. Technical Report MSR-TR-2001-72, Microsoft Research.

Goodman, J. (2001b). Classes for fast maximum entropy training. In *International Conference on Acoustics, Speech, and Signal Processing*, Utah.

Hinton, G. (1986). Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 1–12, Amherst 1986. Lawrence Erlbaum, Hillsdale.

Hinton, G. (2000). Training products of experts by minimizing contrastive divergence. Technical Report GCNU TR 2000-004, Gatsby Unit, University College London.

Jelinek, F. and Mercer, R. L. (1980). Interpolated estimation of Markov source parameters from sparse data. In Gelsema, E. S. and Kanal, L. N., editors, *Pattern Recognition in Practice*. North-Holland, Amsterdam.

Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35(3):400–401.

Miikkulainen, R. and Dyer, M. (1991). Natural language processing with modular neural networks and distributed lexicon. *Cognitive Science*, 15:343–399.

Ney, H. and Kneser, R. (1993). Improved clustering techniques for class-based statistical language modelling. In *European Conference on Speech Communication and Technology (Eurospeech)*, pages 973–976, Berlin.

Niesler, T., Whittaker, E., and Woodland, P. (1998). Comparison of part-of-speech and automatically derived category-based language models for speech recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 177–180.

Pereira, F., Tishby, N., and Lee, L. (1993). Distributional clustering of english words. In *30th Annual Meeting of the Association for Computational Linguistics*, pages 183–190, Columbus, Ohio.

Salton, G. and Buckley, C. (1988). Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523.

Schmidhuber, J. (1996). Sequential neural text compression. *IEEE Transactions on Neural Networks*, 7(1):142–146.

Schutze, H. (1993). Word space. In Giles, C., Hanson, S., and Cowan, J., editors, *Advances in Neural Information Processing Systems 5*, pages pp. 895–902, San Mateo CA. Morgan Kaufmann.

Schwenk, H. (2004). Efficient training of large neural networks for language modeling. In *IEEE joint conference on neural networks*.

Schwenk, H. and Gauvain, J.-L. (2002). Connectionist language modeling for large vocabulary continuous speech recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 765–768, Orlando, Florida.

Stolcke, A. (2002). SRILM - an extensible language modeling toolkit. In *Proceedings of the International Conference on Statistical Language Processing*, Denver, Colorado.

Xu, P., Emami, A., and Jelinek, F. (2003). Training connectionist models for the structured language model. In *Empirical Methods in Natural Language Processing, EMNLP'2003*.

Xu, W. and Rudnicky, A. (2000). Can artificial neural network learn language models. In *International Conference on Statistical Language Processing*, pages M1–13, Beijing, China.

Greedy Spectral Embedding

Marie Ouimet
Dept. IRO, Université de Montréal
P.O. Box 6128, Downtown Branch
Montreal, H3C 3J7, Qc, Canada
ouimema@iro.umontreal.ca

Yoshua Bengio
Dept. IRO, Université de Montréal
P.O. Box 6128, Downtown Branch
Montreal, H3C 3J7, Qc, Canada
bengioy@iro.umontreal.ca

Abstract

Spectral dimensionality reduction methods and spectral clustering methods require computation of the principal eigenvectors of an $n \times n$ matrix where n is the number of examples. Following up on previously proposed techniques to speed-up kernel methods by focusing on a subset of m examples, we study a greedy selection procedure for this subset, based on the feature-space distance between a candidate example and the span of the previously chosen ones. In the case of kernel PCA or spectral clustering this reduces computation to $O(m^2 n)$. For the same computational complexity, we can also compute the feature space projection of the non-selected examples on the subspace spanned by the selected examples, to estimate the embedding function based on all the data, which yields considerably better estimation of the embedding function. This algorithm can be formulated in an online setting and we can bound the error on the approximation of the Gram matrix.

1 Introduction

Many interesting algorithms have been proposed in recent years to perform non-parametric unsupervised learning, either for clustering (spectral clustering) or for manifold learning. Many of these methods can be seen as kernel methods with an adaptive, data-dependent kernel (Bengio et al., 2004). Like other kernel methods, methods such as LLE (Roweis and Saul, 2000), Isomap (Tenenbaum, de Silva and Langford, 2000), kernel Principal Components Analysis (PCA) (Schölkopf, Smola and Müller, 1998), Laplacian Eigenmaps (Belkin and Niyogi, 2003) and spectral clustering (Weiss, 1999; Ng, Jordan and Weiss, 2002), typically require computation of the principal eigenvectors of an $n \times n$ matrix, where n is the number of examples.

In this paper we follow-up on previous work to speed-up kernel methods, such as (Smola and Schölkopf, 2000; Smola and Bartlett, 2001; Williams and Seeger, 2001; Harmeling et al., 2002; Lawrence, Seeger and Herbrich, 2003; Engel, Mannor and Meir, 2003) but focus on spectral methods for **unsupervised learning**. Like in these methods, the main speed-up is obtained by focusing on a subset of the examples, chosen using a greedy selection algorithm. We call the set of selected examples the **dictionary**. Assuming that the kernel is positive semi-definite (which is not always exactly true, but is a good approximation), the above methods are equivalent to a special form of kernel PCA. The criterion used for greedy selection is the distance in feature space between a candidate example and its projection on the subspace spanned by the selected examples. This is a reasonable criterion because the embedding of a new x will be expressed as a linear combination of the $K_D(x, x_i)$, with x_i a dictionary example and K_D the data-dependent kernel associated with the particular method. However, if only the selected examples are used to derive the embedding (i.e. the principal eigenvectors of the feature space covariance matrix), then the resulting embedding will be biased, since the dictionary examples will tend to be distributed more uniformly than the original data (to better cover more directions in feature space). In addition, this projection of the non-dictionary examples can be used to enrich the estimation of the feature space covariance matrix. The resulting algorithm is $O(m^2 n)$, where m is the dictionary size, and requires $O(m^2)$ memory. We show how using a generalized eigen-decomposition algorithm it is possible to take advantage of the sparseness of the kernel. Finally, we also show an efficient way to obtain the kernel normalization required in kernel PCA (additive normalization) and in spectral clustering (divisive normalization).

In section 3 we give more justification as well as the details of this algorithm. We put it in perspective of previously proposed methods in section 4. In section 5 we show with several experiments that the greedy approximation works well and works better than (a) the same algorithm with a randomly selected dictionary (but all the data to estimate the eigenvectors), and (b) using only m random points to estimate the eigenvectors.

2 Spectral Embedding Algorithms and Motivation

Let $D = \{x_1, \ldots, x_n\}$ represent a training set, with x_i a training example. Spectral embedding algorithms (Schölkopf, Smola and Müller, 1998; Weiss, 1999; Roweis and Saul, 2000; Tenenbaum, de Silva and Langford, 2000; Ng, Jordan and Weiss, 2002; Belkin and Niyogi, 2003) provide a vector-valued coordinate for each training example, which would ideally correspond to its coordinate on a manifold near which the data lie. As discussed in (Bengio et al., 2004), spectral embedding methods can generalize the training set embedding to a new example x through the Nyström formula

$$f_k(x) = \frac{\sqrt{n}}{\lambda_k} \sum_{i=1}^{n} v_{ik} K_D(x, x_i) \tag{1}$$

for the k-th embedding coordinate, where K_D is a kernel that is defined using D (hence called data-dependent), and (v_k, λ_k) is the k-th (eigenvector,eigenvalue) pair of the matrix M with entries $M_{ij} = K_D(x_i, x_j)$. Note that this formula reduces to $f_k(x_i) = \sqrt{n} v_{ik}$ for training examples. Depending on the choice of algorithm, the embedding can be further scaled separately in each dimension (e.g. by a factor $\sqrt{\lambda_k}$ for kernel PCA, metric multidimensional scaling (MDS), Isomap, and spectral clustering, and by a factor $\sqrt{n}$ for LLE). Note that K_D is guaranteed to be positive semi-definite for some of these methods (e.g. LLE, kernel PCA, spectral clustering, Laplacian Eigenmaps) but not for others (e.g. Isomap, MDS) but it is usually close to positive semi-definite (e.g. for Isomap the kernel converges to a positive semi-definite one as $n \to \infty$).

Assuming K_D positive semi-definite, there exists a "feature space" $\phi(x)$ in which K_D is a dot product $K_D(x, y) = \phi(x).\phi(y)$. Eq. 1 therefore shows that the function of interest (the embedding of x) can be written as a dot product between $\phi(x)$ and a vector which is a linear combination of the feature space vectors $\phi(x_i)$ associated with the training examples:

$$f_k(x) = \phi(x).(\frac{\sqrt{n}}{\lambda_k} \sum_{i=1}^{n} v_{ik} \phi(x_i)). \tag{2}$$

If we are going to work with a subset of the examples (the dictionary) to define the embedding, we can reduce the above linear combination to one over dictionary examples. There is a simple way to use a dictionary to reduce computation. We first compute the Gram matrix of the dictionary examples and its principal eigenvectors. Then we use the Nyström formula as above to predict the embedding of the other examples. This is essentially the basis for the "Landmark Isomap" algorithm (de Silva and Tenenbaum, 2003), as shown in (Bengio et al., 2004). This is one of the methods that we will evaluate experimentally, and compare to the proposed algorithm, on a Gaussian kernel with additive normalization (corresponding to kernel PCA or metric MDS) and one with divisive normalization (corresponding to spectral clustering or to Laplacian Eigenmaps with Gaussian kernel).

In the two cases that we study here, the data-dependence of the kernel is due to this normalization. The data-dependent kernel K_D is obtained from a data-independent kernel $\tilde{K}$ as follows. For additive normalization (kernel PCA, metric MDS) we have

$$\begin{aligned} K_D(x, y) =& \tilde{K}(x, y) - E_v[\tilde{K}(v, y)] \\ & - E_w[\tilde{K}(x, w)] + E_{v,w}[\tilde{K}(v, w)] \end{aligned} \tag{3}$$

and for divisive normalization (for a positive kernel) we have

$$K_D(x, y) = \frac{\tilde{K}(x, y)}{\sqrt{E_v[\tilde{K}(v, y)] E_w[\tilde{K}(x, w)]}}. \tag{4}$$

In both cases this normalization requires estimation of $E_x[\tilde{K}(x, y)]$ which is normally done with the training set average of the kernel over one of its arguments: $E_x[\tilde{K}(x_i, x)] = \frac{1}{n} \sum_{j=1}^{n} \tilde{K}(x_i, x_j)$. We discuss below how to avoid this computation which would make the whole algorithm running time $O(n^2)$.

3 Proposed Algorithm

Let C denote the covariance matrix of the examples in feature space. One interpretation of the eigenvectors of the Gram matrix is that they correspond to the eigenvectors of C, which can be written as $\frac{1}{\sqrt{\lambda_k}} \sum_{i=1}^{n} v_{ik} \phi(x_i)$. Again, this interpretation is only valid for positive semi-definite kernels, or when working in the subspace associated with non-negative eigenvalues (i.e. with the projection of the $\phi(x)$ on that subspace).

Let $\mathcal{P}$ be the subspace spanned by the dictionary examples in feature space (or the restriction to non-negative eigenvalues). If $\phi(x_t)$ is not far from its projection on $\mathcal{P}$, i.e. it is well approximated by a linear combination of the examples spanning $\mathcal{P}$, than we can replace it by this projection without making a big error. Proposition 1 below formalizes this idea, and Figure 1 shows a geometric interpretation of the projection of $\phi(x_t)$ on the span of the dictionary examples in feature space. Therefore, instead of using a random subset of examples for the dictionary and only using the dictionary to estimate the eigenvectors of C, we propose (1) to select the dictionary examples according to their distance to $\mathcal{P}$, and (2) to use the dictionary examples as well as the projection of the out-of-dictionary examples to estimate the principal eigenvectors of C, thus giving rise to a more precise estimate, almost as good as using the whole data set, according to our experiments. It turns out that (2) comes for free (i.e. for the same cost) once we have accepted to do the computations for (1). Another important consideration is that the selected examples may have statistics that are different from the original examples. For example, if they are chosen as a good "cover" of the training examples, the relative density of dictionary examples will be smaller

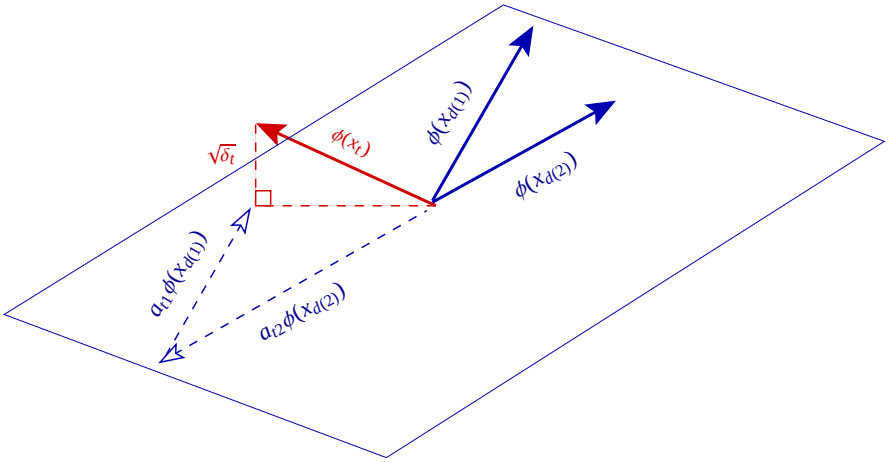


Figure 1: *Out-of-dictionary example $\phi(x_t)$ is approximated in feature space by a linear combination of the dictionary examples $\phi(x_{d(i)})$, $i = 1 \ldots m$.*

Algorithm 1 Greedy Spectral Embedding Algorithm.
Arguments: randomly ordered data set D, tolerance ϵ, embedding dimension p.

1: $n = |D|$, initialize $\mathcal{D} = \{x_1\}$, $M_1 = (K_D(x_1, x_1))$, $M_1^{-1} = \left(K_D(x_1, x_1)^{-1}\right)$.
2: **for** $t = 2$ **to** n **do**
3: **for** $i = 1$ **to** m_{t-1} **do**
4: compute $(k_t(x_t))_i = K_D(x_t, x_{d(i)})$
5: **end for**
6: compute matrix-vector product $\hat{a}_t = M_{t-1}^{-1} k_t(x_t)$
7: compute $\alpha = 1 - \sum_i \hat{a}_{ti}$ and $\beta = \sum_{i,j}(M_{t-1}^{-1})_{ij}$.
8: compute projection weights $a_t = \hat{a}_t + \frac{\alpha}{\beta} M_{t-1}^{-1}[1, \ldots, 1]'$
9: compute projection error $\delta_t = K_D(x_t, x_t) - k_{t-1}(x_t)'\hat{a}_t + \frac{\alpha^2}{\beta}$.
10: **if** $\delta_t > \epsilon$ **then**
11: $m_t = m_{t-1} + 1$, $\mathcal{D} \leftarrow \mathcal{D} \cup \{x_t\}$
12: M_t^{-1} is set according to eq. 9.
13: **else**
14: $m_t = m_{t-1}$, $M_t^{-1} = M_{t-1}^{-1}$
15: **end if**
16: **end for**
17: **for** $i = 1$ **to** m_n **do**
18: **for** $j = 1$ **to** m_n **do**
19: $B_{ij} = A_{it}A_{jt}$ where A_{it} recovered from cache or computed with eq. 10.
20: **end for**
21: **end for**
22: Find the p principal eigenvectors u_k and eigenvalues λ_k of the generalized eigen-problem with left matrix $B = A'A$ and right matrix M_n^{-1}.
23: Embeddings of all examples are given by $v_k = Au_k$ for k-th coordinate.
24: The embedding of a test example x is given by $\frac{f_k(x)}{\sqrt{n}}$, with $f_k(x)$ as in eq. 1.

than the true data density in areas of high density of the original examples. You can observe that phenomenon in the center image of figure 2: the region corresponding to digit 1 is much more dense and contains only 4 dictionary points. Since the directions with high variance are different for the dictionary points, using only this selected subset to form a Gram matrix without projecting the other points would give a completely different embedding.

Computing the distance between $\phi(x_t)$ and the span of the dictionary examples in feature space can be reduced to computations with the kernel, thanks to the kernel trick:

$$\delta_t \doteq \min_{a_t} \|\phi(x_t) - \sum_{i=1}^{m_t} a_{it}\phi(x_{d(i)})\|^2 \tag{5}$$

where $d(i)$ is the index of the i-th dictionary example, and m_t is the size of the dictionary after seeing t examples. We found better results with the constraint $\sum_i a_{it} = 1$. This constraint has the effect of making the solution independent to translations in feature space, since $(\phi(x_t) - b) - \sum_{i=1}^{m_t} a_{it}(\phi(x_{d(i)}) - b) = \phi(x_t) - \sum_{i=1}^{m_t} a_{it}\phi(x_{d(i)})$. In this paper we consider an application of these ideas to an **online** setting (or to reduce memory requirements, at most 2 passes through the data). Let $a_t = (a_{1t}, \ldots, a_{m_t t})$, let M_{t-1} be the Gram matrix of the dictionary examples after seeing $t-1$ examples, and let $k_{t-1}(x)$ be the vector with m_t elements $K_D(x, x_{d(i)})$. The solution for a_t without constraint is

$$\hat{a}_t \doteq M_{t-1}^{-1} k_{t-1}(x_t). \tag{6}$$

The solution with constraint can then be obtained as follows:

$$a_t = \hat{a}_t + \frac{\alpha}{\beta} M_{t-1}^{-1}[1, \ldots, 1]' \tag{7}$$

where $'$ denotes transposition, $\alpha = 1 - \sum_i \hat{a}_{ti}$ and $\beta = \sum_{i,j}(M_{t-1}^{-1})_{ij}$. The corresponding value of the projection distance is then obtained as follows:

$$\delta_t = \hat{\delta}_t + \frac{\alpha^2}{\beta}, \tag{8}$$

where $\hat{\delta}_t = K_D(x_t, x_t) - k_{t-1}(x_t)'\hat{a}_t$. We introduce the parameter ϵ that controls the accuracy of the approximation. x_t is added to the dictionary if $\delta_t > \epsilon$, which means that x_t is far from its projection on $\mathcal{P}$ in feature space. When a point is added to the dictionary, M_t^{-1} needs to be computed. To do so efficiently (in $O(m_t^2)$ computations), we can take advantage of our knowledge of M_{t-1}^{-1} and the matrix inversion lemma:

$$M_t^{-1} = \frac{1}{\hat{\delta}_t}\begin{pmatrix} \hat{\delta}_t M_{t-1}^{-1} + \hat{a}_t\hat{a}_t' & -\hat{a}_t \\ -\hat{a}_t' & 1 \end{pmatrix} \quad (9)$$

following the recursive update approach in (Engel, Mannor and Meir, 2003) for kernel regression.

After we have selected the dictionary, we want to compute the projection of all the examples on its span. This will be different from the projection computed online in eq. 7 because it will use the final dictionary and M_n^{-1} instead of M_{t-1}^{-1}:

$$\begin{aligned} \hat{A}_t &\doteq M_n^{-1} k_n(x_t) \\ A_t &\doteq \hat{A}_t + \frac{1 - \sum_i \hat{A}_{ti}}{\sum_{i,j}(M_n^{-1})_{ij}} M_n^{-1}[1, \ldots, 1]'. \end{aligned} \quad (10)$$

Let A denote the matrix with rows A_t. It can be shown that the complete Gram matrix is approximated by AM_nA'. We can bound the error on each entry of the Gram matrix by the same parameter we used to select dictionary examples:

Proposition 1

For all $x_t, x_u \in D$, $|K_D(x_t, x_u) - (AM_nA')_{tu}| \leq \epsilon$.

Proof: Let $r_t = \phi(x_t) - \sum_{i=1}^{m_n} a_{it}\phi(x_{d(i)})$. Notice that if $x_t \in \mathcal{D}$, then $\|r_t\| = 0$, otherwise, $\|r_t\| = \sqrt{\delta_t}$ and r_t is orthogonal to the span of $\phi(x_{d(i)})$, $i = 1, ..., m_n$. Then we have

$$\begin{aligned} &K_D(x_t, x_u) = \phi(x_t).\phi(x_u) \\ &= (r_t + \textstyle\sum_{i=1}^{m_n} a_{it}\phi(x_{d(i)})).(r_u + \sum_{i=1}^{m_n} a_{iu}\phi(x_{d(i)})) \\ &= r_t.r_u + r_t.\textstyle\sum_{i=1}^{m_n} a_{iu}\phi(x_{d(i)}) + r_u.\sum_{i=1}^{m_n} a_{it}\phi(x_{d(i)}) \\ &\quad + (\textstyle\sum_{i=1}^{m_n} a_{it}\phi(x_{d(i)})).(\sum_{i=1}^{m_n} a_{iu}\phi(x_{d(i)})) \\ &= r_t.r_u + (\textstyle\sum_{i=1}^{m_n} a_{it}\phi(x_{d(i)})).(\sum_{i=1}^{m_n} a_{iu}\phi(x_{d(i)})) \\ &= r_t.r_u + (AM_nA')_{tu}. \end{aligned}$$

So we have

$$\begin{aligned} &|K_D(x_t, x_u) - (AM_nA')_{tu}| \\ &= |r_t.r_u| \leq \sqrt{\delta_t}\sqrt{\delta_u} \leq \epsilon. \end{aligned}$$

The eigenvectors of AM_nA' would give us the embedding of all the training examples, but this is an $n \times n$ matrix. We want to take advantage of its factorization. Note that if u_k is an eigenvector of $M_nA'A$ then $v_k = Au_k$ is an eigenvector of AM_nA' with the same eigen values. Unfortunately, $M_nA'A$ is not symmetric, But we can still compute the eigenvectors efficiently in time $O(m^3)$, by solving the $m \times m$ generalized eigen-system $Lu = \lambda Ru$ with left matrix $L = A'A$ and right matrix $R = M_n^{-1}$ (which we already have from our recursive updates). Finally this gives rise to algorithm 1.

3.1 Computational Cost and Memory Usage

The expensive steps of the algorithm 1 are step 6 ($O(m_{t-1}^2)$ per step, $O(nm^2)$ overall), step 12 ($O(m_{t-1}^2)$ per step, $O(nm^2)$ overall), step 19 ($O(nm^2)$), step 22 ($O(m^3)$), and step 23 ($O(pnm)$). The memory usage is $O(m^2)$ to store M_t (which becomes M_n at the end), if the a_{it} are recomputed in step 23, or $O(nm)$ to store A if they are not recomputed (depending on how large n is and how much memory is available). Hence time is $O(nm^2)$ and memory either $O(m^2)$ or $O(nm)$ (time-memory trade-off only in the constant factor for time).

Algorithm 2 Constructs a dictionary of specified size m_n with an almost minimal level of error.
Arguments: randomly ordered data set D, wanted subset size m_n.

1: Initialize $\mathcal{D} = \{x_1\}$, $M_t = (K_D(x_1, x_1))$, $M_t^{-1} = \left(K_D(x_1, x_1)^{-1}\right)$.
2: Set $\epsilon^+ = 0$ or something for $m_n^+ > m_n$
3: Set $\epsilon^- = 1$ or something for $m_n^- < m_n$
4: Check that $m_n^- < m_n$ by running steps 2 to 16 of algo. 1 using $\epsilon^-, \mathcal{D}, M_t, M_t^{-1}$ and data set $D - \mathcal{D}$, stopping if $m_t^- > m_n$. Store the obtained dictionary and corresponding matrices in $\mathcal{D}^-, M_t^-, M_t^{-1}$.
5: **if** $m_t^- = m_n$ **then**
6: **return** $\mathcal{D}^-, M_t^-, M_t^{-1}$
7: **else if** $m_t^- > m_n$ **then**
8: $\epsilon^+ = \epsilon^-$
9: $\epsilon^- = 10\epsilon^-$
10: **goto** step 4
11: **else**
12: Set $\mathcal{D} = \mathcal{D}^-, M_t = M_t^-, M_t^{-1} = M_t^{-1}$
13: **end if**
14: **for** $i = 1$ **to** 40 **do**
15: Set $\epsilon' = \epsilon^+ + (\epsilon^- - \epsilon^+)/2$
16: Run steps 2 to 16 of algo. 1 using $\epsilon', \mathcal{D}, M_t, M_t^{-1}$ and data set $D - \mathcal{D}$, stopping if $m_t' > m_n$. Store the obtained dictionary and corresponding matrices in $\mathcal{D}', M_t', M_t'^{-1}$.
17: **if** $m_t' = m_n$ **then**
18: **return** $\mathcal{D}', M_t', M_t'^{-1}$
19: **else if** $m_t' > m_n$ **then**
20: Set $\epsilon^+ = \epsilon'$
21: **else**
22: Set $\epsilon^- = \epsilon', \mathcal{D} = \mathcal{D}', M_t = M_t', M_t^{-1} = M_t'^{-1}$
23: **end if**
24: **end for**
25: Run steps 2 to 16 of algo. 1 using $\epsilon^+, \mathcal{D}, M_t, M_t^{-1}$ and data set $D - \mathcal{D}$, stopping when $m_t^+ = m_n$. Store the obtained dictionary and corresponding matrices in $\mathcal{D}^+, M_t^+, M_t^{+-1}$.
26: **return** $\mathcal{D}^+, M_t^+, M_t^{+-1}$

Although the goal was to get an online algorithm, it is also possible to formulate the method in a setting more similar

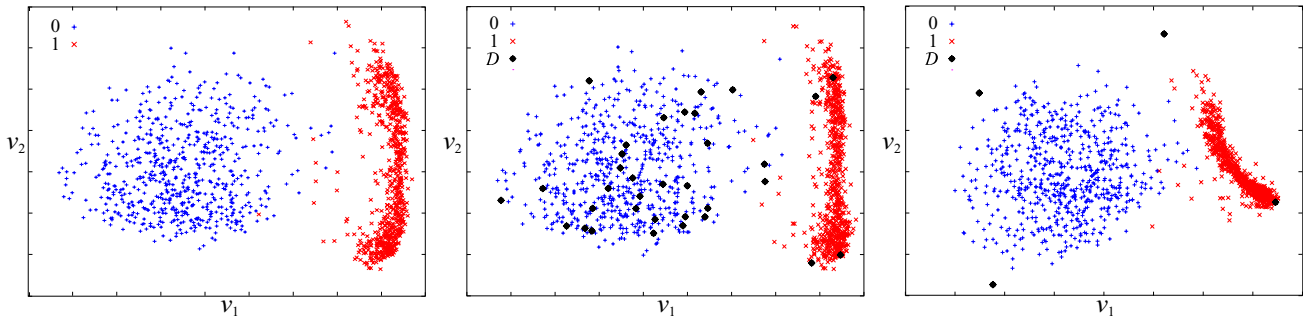


Figure 2: First two dimensions of the embeddings obtained with kernel PCA on classes 0 (+) and 1 (×) of the MNIST data set ($\sigma = 31.6$, $n = 1300$). The dictionary points are noted by ◆. The image on the left is the embedding obtained with the full Gram matrix. With 34 points in the dictionary, we obtain an almost identical embedding. Even with only 4 points, the embedding is reasonable and similar.

to previously proposed methods where instead of specifying the accuracy parameter ϵ, we specify the subset size m. To do so, one needs to first search for a good value of ϵ before applying algorithm 1. This can also be done in $O(nm^2)$ operations and $O(m^2)$ in memory usage. Algorithm 2 constructs a dictionary of the wanted size with an almost minimal error level. Beginning with an ϵ that gives a too small dictionary and one giving a too large dictionary, it does a binary search for the right ϵ while always keeping small dictionaries and only adding points to it. It first adds to the dictionary the points that are the farthest from $\mathcal{P}$ and gradually adds points that are closer and closer. This ensures that the error level will be near optimal. One gets the desired embedding by running steps 17 to 24 of algorithm 1 with the obtained dictionary and corresponding matrices.

3.2 Additive and Divisive Normalizations

We have not discussed how to perform additive or divisive normalization yet. If we compute the usual complete training set averages to estimate $E_x[\tilde{K}(x,y)]$, then the whole algorithm becomes $O(n^2)$ because we have to compute the values of the data-independent kernel $\tilde{K}(x_i, x_j)$ for all data pairs. There are two solutions to this problem, which gave similar results in our experiments. One is to use a random subset of the examples (or the dictionary examples) to estimate these averages. If we average over less than m^2 examples the overall algorithm would remain $O(nm^2)$. The other solution is to compute the exact normalization associated with the implicit estimated Gram matrix AM_nA'.

Proposition 2

Additive normalization of AM_nA' can be obtained by using instead of AM_nA' the Gram matrix $\tilde{A}M_n\tilde{A}'$ with $\tilde{A} = A - B$, and B the matrix whose rows are all identical and equal to the average row of A, $E_t[A_t]$. Similarly, divisive normalization can be obtained by using the Gram matrix $\tilde{A}M_n\tilde{A}'$ with $\tilde{A}_{i\cdot} = \frac{A_{i\cdot}}{\sqrt{nA_{i\cdot}(M_nE_t[A_t])}}$.

The proof is straightforward comparing $\tilde{A}_iM_n\tilde{A}_j{}'$ to $K_D(x_i, x_j)$ in eq. 3 and 4. Using this proposition, the algorithm remains the same except that the matrix A is replaced by $\tilde{A}$ in the last steps (22 and 23). Under constraint $\sum_i a_{it} = 1$, additive normalization before or after the computation of A made no empirical difference, since the approximation is invariant to a translation in feature space. The only difference is the number of points over which the averages are made. This is not the case for the divisive normalization, where if we want to compute the approximation in the feature space induced by the normalized kernel, the normalization should be applied before, using a subset of the examples. For this normalization, the constraint $\sum_i a_{it} = 1$ does not help and could be removed, but it did not make a measurable difference in the experiments.

4 Related work

Other sparse greedy kernel methods in $O(m^2n)$ have been proposed. (Smola and Bartlett, 2001) and (Lawrence, Seeger and Herbrich, 2003) present algorithms for greedy Gaussian processes. The main difference with our method is that they do m passes on the whole data set (or a random subset of it to reduce computation) and each time they select the example that improves the most a global criterion. They also use a recursive update for inverting matrices. In (Smola and Schölkopf, 2000) the setting is very general, aiming to approximate a matrix K by $\tilde{K}$, a lower rank matrix, by minimizing the Frobenius norm of the residual $K - \tilde{K}$. This is again done with m passes to select the best basis function among a random subset of all the $n - m_t$ function. Our algorithm is different from these methods by requiring at most 2 passes on the data set. Instead, the approximation we use is close to the approach presented in (Engel, Mannor and Meir, 2003) where it was used in a sequential and supervised setting to perform Kernel RLS. (Williams and Seeger, 2001) suggests to compute the eigen-decomposition on a subset of m examples and then use the Nyström formula to get an approximation of the eigen-decomposition for the $n - m$ other points. They argue that their method, although less precise than (Smola and Schölkopf, 2000), is much faster. In our experiments, we will compare our greedy technique to this

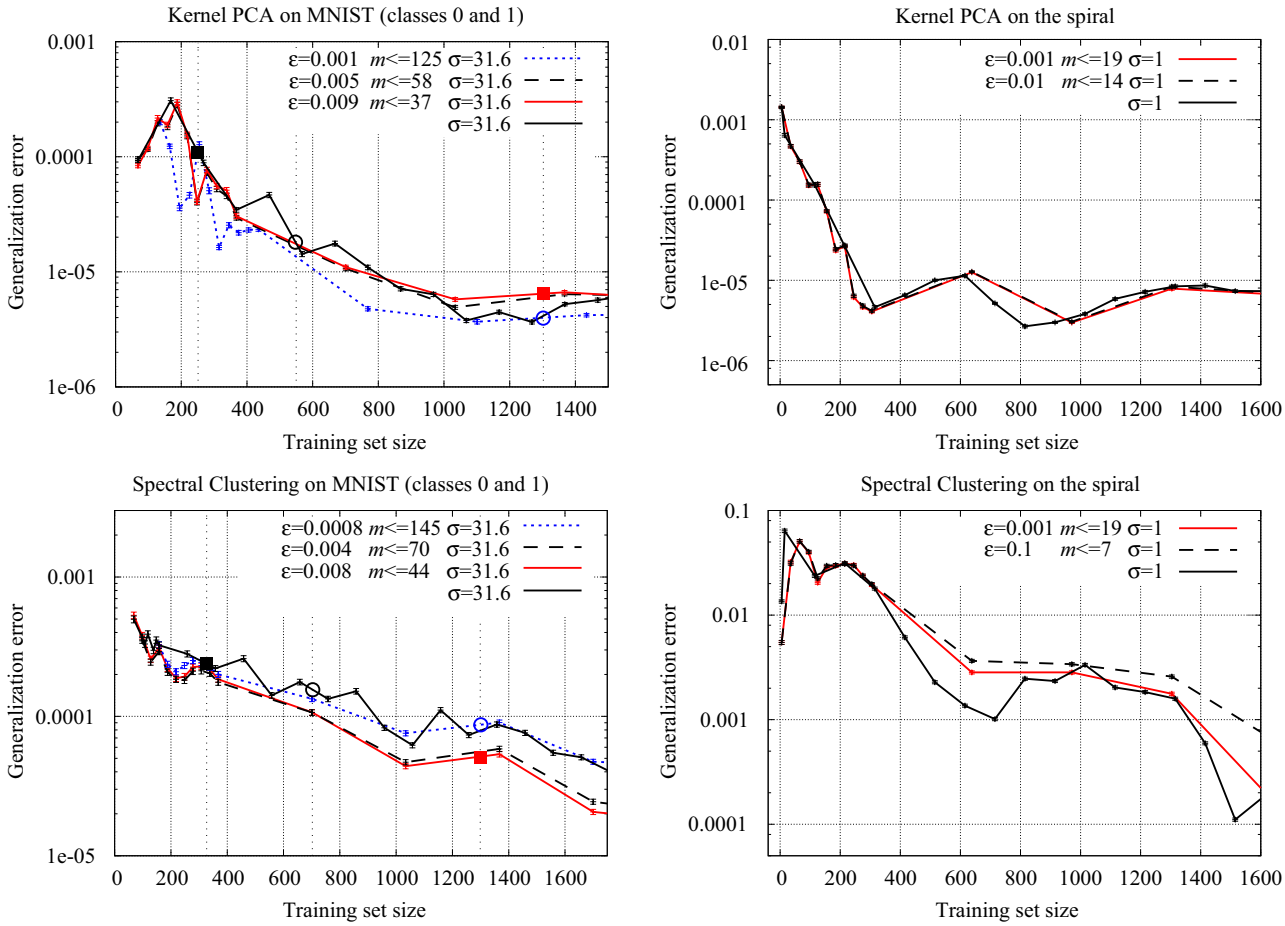


Figure 3: Even for really small dictionary sizes, the generalization performance is comparable to the one obtained with the Gram matrix of the whole training set. Curves corresponding to the dictionary methods are indexed with σ, the standard deviation of the Gaussian kernel, ϵ, the accuracy parameter and m, the resulting maximum number of points in the dictionary (obtained for the maximum n). The curves corresponding to non-dictionary methods are indexed with σ only. There are 3 principal components for MNIST and 2 for the spiral. The individual experiments associated with the symbols ■ and ■ have the same running time, but the generalization performance is far better for the dictionary method. We observe the same phenomenon for symbols ⚬ and ⚬ corresponding to a longer running time. More details on the relationship between running time and generalization error are given in table 1.

Nyström method showing that for the same computational time we get better performance. Another related method is (Harmeling et al., 2002) in which we search for the largest random subset of examples for which the Gram matrix is of full rank and project all the examples on this subset. This technique is more computationally expensive than the greedy selection since it requires to do several matrix decompositions to compute the rank. We will also see in the experimental section that random subset selection leads to larger subsets for the same error level.

5 Experimental Evaluation

The new algorithm was evaluated on two data sets: a 2-D artificially generated spiral, and images of the digits 0 and 1 from the MNIST data set (scaled down from 28×28 to 14×14). Two embedding methods were compared and evaluated out-of-sample using the Nyström formula: spectral clustering and kernel PCA, both with the Gaussian kernel, with different values of standard deviation (shown in Figure 3). An example of the embeddings obtained with the dictionary method on the MNIST data set is given in figure 2.

The goals of the experiments were (1) to verify that the algorithm worked (in the sense of giving an embedding close to the one obtained when training with the whole data set) and (2) to verify that it worked better than (a) the same algorithm with randomly selected dictionary (Harmeling et al., 2002) and (b) training on a small subset and generalization with the Nyström formula (Williams and Seeger, 2001; Bengio et al., 2004). To compare the embeddings given by these different methods, we will need a reference embedding. We first divide our data set in three parts: D_1,

Table 1: Comparison between generalization errors for the dictionary and non-dictionary methods at a fixed running time. The errors and the n corresponds to the ones in Figure 3 for MNIST. Some of these results are plotted in Figure 3 with square and circle symbols. Similar results are obtained for a fixed memory space.

	WITHOUT DICTIONARY		**WITH DICTIONARY**	
TIME KPCA (s)	**MAX. SIZE**	**GEN. ERROR** ±95% **C.I.**	**MAX. SIZE**	**GEN. ERROR** ±95% **C.I.**
05.71	$n = 250$	$1.48e^{-4} \pm 8.9e^{-6}$	$n = 1300,\ m = 34$	$6.64e^{-6} \pm 2.3e^{-7}$
05.72	$n = 250$	$1.48e^{-4} \pm 8.9e^{-6}$	$n = 1300,\ m = 55$	$6.40e^{-6} \pm 2.1e^{-7}$
06.46	$n = 550$	$8.27e^{-6} \pm 4.0e^{-7}$	$n = 1300,\ m = 126$	$3.99e^{-6} \pm 1.5e^{-7}$
14.02	$n = 1300$	$3.47e^{-6} \pm 1.4e^{-7}$		
TIME SC (s)				
05.80	$n = 325$	$2.41e^{-4} \pm 1.2e^{-5}$	$n = 1300,\ m = 41$	$5.16e^{-5} \pm 2.3e^{-6}$
05.95	$n = 400$	$2.37e^{-4} \pm 1.2e^{-5}$	$n = 1300,\ m = 65$	$5.61e^{-5} \pm 2.5e^{-6}$
07.18	$n = 700$	$1.58e^{-4} \pm 7.3e^{-6}$	$n = 1300,\ m = 138$	$8.74e^{-5} \pm 3.8e^{-6}$
14.06	$n = 1300$	$7.93e^{-5} \pm 3.4e^{-6}$		

D_2, D_3, where D_2 and D_3 are large. We compute a reference embedding by applying standard kernel PCA or spectral clustering to $D_2 \cup D_3$ and keeping the part corresponding to D_2. We then train the methods we want to compare on D_1. Using the obtained eigenvectors, we compute the out-of-sample embedding for every element of the test set D_2 with the Nyström formula. We will compare this embedding to the reference embedding of D_2 by aligning them with a simple linear regression (that maps each example's coordinate in one embedding with the same example's coordinate in the other embedding). In our experiments, the reported generalization error is the average squared difference between the corresponding points of these aligned embeddings. We also show the 95% confidence intervals associated to the standard errors of these averages. For the spiral data, we used sets of sizes $|D_1| \leq 3333$, $|D_2| = 3330$ and $|D_3| = 3332$. For the MNIST data, $|D_1| \leq 3365$, $|D_2| = 3267$ and $|D_3| = 3332$.

In the experiments shown in Figure 3, we verify that the greedy algorithm works about as well as when we use the full Gram matrix. On the horizontal axis, we vary the size of the training set D_1 and on the vertical axis, we show the out-of-sample errors obtained with the Nyström formula. The dictionary method (curves indexed with ϵ, m, σ) is trained with fixed values of ϵ, yielding different subset sizes as the training set grows. The largest subset size obtained is shown as "$m \leq ...$". To compare, we compute the eigenvectors of the full Gram matrix corresponding to same training set (ordinary kernel PCA and spectral clustering). The corresponding generalization errors are reported by the curves indexed with σ only (full black curves). The main feature to note is that for a training set of size n, the results with the dictionary of size $m \ll n$ are very close to the results using ordinary training with all the examples, i.e. the greedy algorithm generalizes about as well as full training. The other important conclusion from this figure is that the dictionary method works much better than the simple Nyström method with eigenvectors estimated on a random dictionary (compare for example the error of the full Gram matrix method trained with a dataset of size $n = 125$ with the dictionary method for $n = 1400$ and $m = 125$ for kernel PCA on MNIST). To be more fair, one can compare the Nyström method to the dictionary method at equal running time instead of equal subset size. Table 1 shows that for the same running time, the dictionary method yields quite smaller generalization error. In figure 3 and in table 1 you can notice a slight increase in error as the dictionary grows for spectral clustering on the MNIST data set, but as expected, the curve for the larger dictionary is closer to the one for the whole dataset. This increase in error is probably due to overfitting; using a small dictionary is a way to regularize. Finally, in Figure 4 we compare the proposed selection algorithm vs a random selection procedure. In both cases we project the other points on the span of the subset in feature space. We also show the performance of the Nyström technique with the eigen-decomposition made on a random subset of points, ignoring the other points of the training set. In these experiments we used a training set of size 3365 and show the test set error for different values of subset size m. The first thing we notice is that projecting the other examples makes a huge difference. For this data set, we need a subset of size about 900 for the Nyström technique to achieve the performance we get with less than 100 points in the subset if we project the 3265 other points. Although the greedy selection and the random selection behave the same for subsets of size 65 and more, this is not the case for smaller sizes. The greedy selection strategy reduces generalization error significantly, the more so for smaller dictionaries, as expected. Note that for this problem a greedy dictionary of size 44 gave performances similar to when the complete Gram matrix was used, as shown in figure 3, but figure 4 suggests that a dictionary of size about 22 would have been enough. For these dictionary sizes, the performance of the random dictionary method is not as good and consequently, an approach like (Harmeling et al., 2002) will need to pick a subset of size at least 65 to reach about the same error level.

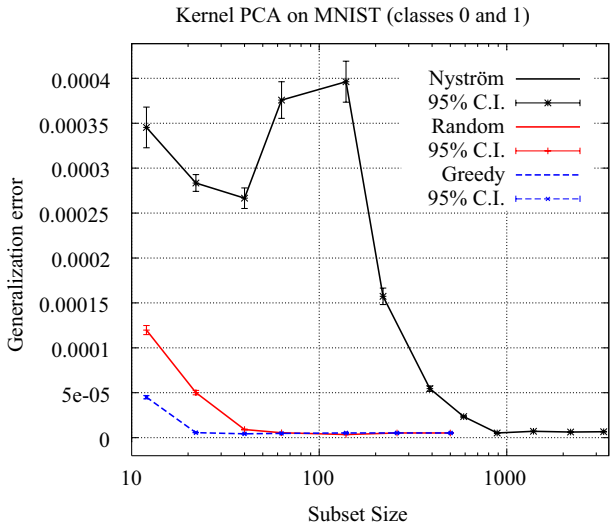


Figure 4: Comparison of the dictionary method vs the Nyström method where the eigen-decomposition is made only on a random subsets of the training set. We show two subset selection strategies for the dictionary method: the proposed greedy selection and a simple random selection. These experiments were performed with a training set of size 3365, with 3 principal components.

6 Conclusion

Spectral embedding methods are useful for manifold learning (non-linear dimensionality reduction) and clustering (spectral clustering) but require an expensive $O(n^3)$ operation with $O(n^2)$ memory requirement, with n the number of examples. In this paper we propose an efficient algorithm that yields almost as good results and reduces computation to $O(nm^2)$ and memory to $O(m^2)$ where m is the size of a small subset of selected examples (the dictionary). The algorithm selects examples greedily and sequentially based on their distance to the span of the already selected ones, in the kernel feature space. It then uses the projection of all n examples on that span to estimate the required eigenvectors. We show how to formulate kernel PCA and spectral clustering within this framework. In theory one can also write a functional form for other data dependant kernels like LLE and Isomap kernels, but the update of the matrix becomes relatively ineffective in these cases and should be the subject of future research. Experiments show that for kernel PCA and spectral clustering, the selection method is significantly better than random selection, and better than simply using the Nyström method to generalize (as in Isomap's landmark variant (de Silva and Tenenbaum, 2003)). In fact it worked almost as well as training with all the data.

Acknowledgements

The authors would like to thank the following funding organizations for support: NSERC, FQRNT, MITACS, IRIS, and the Canada Research Chairs.

References

Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396.

Bengio, Y., Delalleau, O., Le Roux, N., Paiement, J.-F., Vincent, P., and Ouimet, M. (2004). Learning eigenfunctions links spectral embedding and kernel PCA. *Neural Computation*, to appear.

de Silva, V. and Tenenbaum, J. (2003). Global versus local methods in nonlinear dimensionality reduction. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 705–712, Cambridge, MA. MIT Press.

Engel, Y., Mannor, S., and Meir, R. (2003). The kernel recursive least squares algorithm. Technical Report submitted to Trans. Sig. Proc., MIT.

Harmeling, S., Ziehe, A., Kawanabe, M., and Müller, K.-R. (2002). Kernel feature spaces and nonlinear blind souce separation. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press.

Lawrence, N., Seeger, M., and Herbrich, R. (2003). Fast sparse gaussian process methods: The informative vector machine. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 609–616. MIT Press.

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: analysis and an algorithm. In Dietterich, T., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press.

Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319.

Smola, A. and Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. In Langley, P., editor, *International Conference on Machine Learning*, pages 911–918, San Francisco. Morgan Kaufmann.

Smola, A. J. and Bartlett, P. (2001). Sparse greedy gaussian process regression. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*.

Tenenbaum, J., de Silva, V., and Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

Weiss, Y. (1999). Segmentation using eigenvectors: a unifying view. In *Proceedings IEEE International Conference on Computer Vision*, pages 975–982.

Williams, C. K. I. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 682–688, Cambridge, MA. MIT Press.

FastMap, MetricMap, and Landmark MDS are all Nyström Algorithms

John C. Platt
Microsoft Research
1 Microsoft Way
jplatt@microsoft.com

Abstract

This paper unifies the mathematical foundation of three multidimensional scaling algorithms: FastMap, MetricMap, and Landmark MDS (LMDS). All three algorithms are based on the Nyström approximation of the eigenvectors and eigenvalues of a matrix. LMDS is applies the basic Nyström approximation, while FastMap and MetricMap use generalizations of Nyström, including deflation and using more points to establish an embedding. Empirical experiments on the Reuters and Corel Image Features data sets show that the basic Nyström approximation outperforms these generalizations: LMDS is more accurate than FastMap and MetricMap with roughly the same computation and can become even more accurate if allowed to be slower.

1 INTRODUCTION

Multidimensional Scaling (MDS) [4] is an important method for visualizing and processing high-dimensional or graphical data. MDS takes as input a distance matrix between items. It produces a coordinate vector for each item in a Euclidean space whose dimension is user-specifiable. This process is known as *embedding*.

MDS is applicable to two different tasks: 1) dimensionality reduction, which measures a set of distances between items, then applies MDS to the resulting (perhaps sparse) distance matrix; and 2) converting graphs to vectors, where a data set is expressed as a set of similarity relationships between items that is then converted into a simple table of vectors.

The original MDS algorithms from the 1960s [4] are not appropriate for large scale applications because they require an entire $N \times N$ distance matrix to be stored in memory and may have $O(N^3)$ complexity. However, in the last 10 years, several scalable MDS algorithms have been proposed. For example, Falutsos and Lin [7] proposed FastMap, which is an MDS method that determines one coordinate at a time by examining a constant number of rows of the distance matrix. Wang, et. al [12] proposed an improvement on FastMap, called MetricMap, which attempts to do the entire projection at once. de Silva and Tennenbaum [5] proposed Landmark MDS (LMDS) as another attempt at scalable MDS.

FastMap, MetricMap, and LMDS are all classical MDS algorithms that start with a distance matrix assumed to have been computed from points in a low-dimensional space. They then (approximately) minimize a quadratic cost function between the resulting embedding coordinates and the original (hidden) coordinates that created the distance matrix. Another set of MDS algorithms are based on spring models [3, 14]. These spring models minimize a cost function that is non-quadratic in the coordinates: the squared difference between embedded and observed distances between items. Recent work has also accelerated these spring models, but they are prone to local minima [3]. Spring models are not considered in this paper.

The proliferation of MDS algorithms may lead to frustration amongst practitioners because it is unclear how the algorithms relate to one another or how they compare against each other. This paper attempts to clarify the situation in two ways. First, this paper shows that FastMap, MetricMap, and LMDS are all algorithms based on the Nyström approximation of the eigenvectors of a large matrix [1, 13], based only on a rectangular sub-matrix of the large matrix. LMDS uses the basic Nyström approximation, FastMap uses Nyström to find one eigenvector at a time, and MetricMap uses an irregular sub-matrix to find the eigenvectors. The paper presents empirical comparisons between FastMap, MetricMap, and LMDS, to determine which variation

of Nyström is optimal.

2 MATHEMATICAL BACKGROUND

This section presents a step-by-step introduction to Classical MDS and the Nyström eigenvector approximation. The LMDS algorithm is then derived based on the combination of the two, as first described in [2].

2.1 REVIEW OF CLASSICAL MDS

FastMap, MetricMap, and LMDS are all multi-dimensional scaling (MDS) algorithms [4] that map a matrix of dissimilarities $\mathbf{D}$ between N items to a k-dimensional coordinate vector for each item ($\vec{x}_i$). This paper's derivation for all three of these algorithms starts with metric MDS, which assumes that the entries in the dissimilarities matrix are Euclidean distances. These three algorithms then utilize classical MDS, which chooses the k-dimensional coordinates to minimize the squared difference between the distances in the embedded and the original spaces.

Classical MDS proceeds in two steps. First, the distance matrix $\mathbf{D}$ undergoes "double-centering" to convert it from a distance matrix to a new matrix $\mathbf{K}$:

$$K_{ij} = -\frac{1}{2}\left(D_{ij}^2 - e_j \sum_i c_i D_{ij}^2 - e_i \sum_j c_j D_{ij}^2 + \sum_{i,j} c_i c_j D_{ij}^2\right), \quad (1)$$

where $\sum_i c_i = 1$ and e_i is the vector of all ones. If the original distance matrix $\mathbf{D}$ is a Euclidean distance matrix in d-dimensional space, then $\mathbf{K}$ is a matrix of dot products between coordinate vectors in that same space [11]. This is also known as a Gram or kernel matrix. If the original distance matrix is Euclidean, then $\mathbf{K}$ is symmetric and positive semi-definite. The parameters c_i in (1) determine the origin of the coordinate vectors (which is not constrained by the distance matrix $\mathbf{D}$).

The second step of classical MDS is to extract the coordinate vectors from the kernel matrix $\mathbf{K}$ through eigenvector decomposition. If the matrix $\mathbf{D}$ is symmetric, then $\mathbf{K}$ is symmetric and can be decomposed into

$$\mathbf{K} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T \quad (2)$$

where $\mathbf{Q}$ is a matrix whose columns are orthonormal eigenvectors and $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues. The k-dimensional coordinate vectors that would give rise to the kernel matrix $\mathbf{K}$ are the scaled rows of $\mathbf{Q}$. In order to minimize the difference between the embedded and original distances with a fixed number of dimensions k, the eigenvectors with the top k eigenvalues are retained. Assuming that the eigenvalues are ordered by decreasing eigenvalue, the jth component of point i's coordinate vector is:

$$x_{ij} = \sqrt{\lambda_j} Q_{ij}, \quad (3)$$

where λ_j is the jth eigenvalue and here the index j runs only from 1 to k, rather than to N (the size of $\mathbf{K}$).

2.2 THE NYSTRÖM APPROXIMATION

There are numerous ways of performing the decomposition (2). If only the top k eigenvectors are needed, then orthogonal iteration or Lanczos iteration can be applied [8].

However, the three MDS algorithms that are the subject of this paper use an approximation method from physics, called the Nyström approximation [1, 13]. To use Nyström, first choose m items in the distance and kernel matrix at random. Without loss of generality, permute the m items to be the first rows and columns of these matrices. The $\mathbf{K}$ and $\mathbf{D}$ can then be partitioned into submatrices:

$$\mathbf{K} = \begin{array}{|c|c|}\hline \mathbf{A} & \mathbf{B} \\ \hline \mathbf{B}^T & \mathbf{C} \\ \hline \end{array}, \quad \mathbf{D} = \begin{array}{|c|c|}\hline \mathbf{E} & \mathbf{F} \\ \hline \mathbf{F}^T & \mathbf{G} \\ \hline \end{array}, \quad (4)$$

where $\mathbf{A}$ and $\mathbf{E}$ have dimension $m \times m$; $\mathbf{B}$ and $\mathbf{F}$ have dimension $m \times (N-m)$; and $\mathbf{C}$ and $\mathbf{G}$ have dimension $(N-m) \times (N-m)$.

The Nyström approximation permits the computation of the coordinates $\vec{x}_i$ using only the information in matrices $\mathbf{A}$ and $\mathbf{B}$. Nyström assumes that $\mathbf{K}$ is positive semi-definite, and hence a Gram matrix. Thus, $\mathbf{K}$ should be expressible in terms of dot products between columns of matrices $\mathbf{X}$ and $\mathbf{Y}$ [1]:

$$\mathbf{K} = \begin{bmatrix} \mathbf{X}^T\mathbf{X} & \mathbf{X}^T\mathbf{Y} \\ \mathbf{Y}^T\mathbf{X} & \mathbf{Y}^T\mathbf{Y} \end{bmatrix}. \quad (5)$$

Identifying the submatrices in (5) with those in $\mathbf{K}$ in (4) yields

$$\begin{aligned} \mathbf{A} &= \mathbf{X}^T\mathbf{X}, \\ \mathbf{B} &= \mathbf{X}^T\mathbf{Y}. \end{aligned} \quad (6)$$

The first equation in (6) is standard in classical MDS: the solution for $\mathbf{X}$ is to eigendecompose $\mathbf{A}$:

$$\mathbf{A} = \mathbf{U}\mathbf{\Gamma}\mathbf{U}^T, \quad (7)$$

and then assign the coordinates

$$\mathbf{X} = \mathbf{\Gamma}_{[k]}^{1/2}\mathbf{U}_{[k]}^T, \quad (8)$$

where the subscript $[k]$ indicates the submatrices corresponding to the eigenvectors with the k largest positive eigenvalues. The coordinates corresponding to $\mathbf{B}$ can be derived by solving the linear system:

$$\mathbf{Y} = \mathbf{X}^{-T}\mathbf{B} = \mathbf{\Gamma}_{[k]}^{-1/2}\mathbf{U}_{[k]}^{T}\mathbf{B}. \tag{9}$$

Combining equations (8) and (9) together and writing the coordinate as rows in a matrix results in

$$x_{ij} = \begin{cases} \sqrt{\gamma_j}U_{ij} & \text{if } i \le m; \\ \sum_p B_{pi}U_{pj}/\sqrt{\gamma_j} & \text{otherwise,} \end{cases} \tag{10}$$

where U_{ij} is the ith component of the jth eigenvector of $\mathbf{A}$ and γ_j is the jth eigenvalue of $\mathbf{A}$. As in (3) the j index only runs from 1 to k, in order to make a k-dimensional embedding.

The Nyström approximation can be understood by plugging (8) and (9) back into (6). Nyström approximates the full matrix $\mathbf{K}$ by

$$\tilde{\mathbf{K}} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B} \end{bmatrix}. \tag{11}$$

This approximation is exact when $\mathbf{K}$ is of rank m or less. The quality of the approximation is proportional to $||\mathbf{C} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}||$.

To turn Nyström into an MDS method, matrices $\mathbf{A}$ and $\mathbf{B}$ must be derived only from submatrices $\mathbf{E}$ and $\mathbf{F}$ (from $\mathbf{D}$). This can be done by choosing the centering coefficients in equation (1) to be

$$c_i = \begin{cases} 1/m & \text{if } i \le m; \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

which yields the centering formulas

$$\begin{aligned} A_{ij} &= -\frac{1}{2}\left(E_{ij}^2 - e_i\frac{1}{m}\sum_p E_{pj}^2 \right. \\ &\quad \left. -e_j\frac{1}{m}\sum_q E_{iq}^2 + \frac{1}{m^2}\sum_{p,q} E_{pq}^2\right) \end{aligned} \tag{13}$$

$$B_{ij} = -\frac{1}{2}\left(F_{ij}^2 - e_i\frac{1}{m}\sum_p F_{qj}^2 - e_j\frac{1}{m}\sum_p E_{ip}^2\right), \tag{14}$$

where the constant centering term in (14) is dropped, because it introduces an irrelevant shift of origin.

Note that Nyström can also be used to approximately solve spectral clustering problems [1], by using a normalized graph Laplacian, instead of a centering matrix, to transform a data matrix into a kernel matrix.

2.3 LMDS

The combination of equations (13) and (14), followed by (7), then (10) is almost the LMDS algorithm [5]. LMDS is so named because Classical MDS is first applied to a subset of the points (in $\mathbf{A}$), called "landmarks."

The only difference between the algorithm derived here and LMDS is the centering formula (14). In [5], the centering formula

$$B_{ij} = -\frac{1}{2}\left(F_{ij}^2 - e_i\frac{1}{m}\sum_p E_{ip}^2\right) \tag{15}$$

is used. This simplification does not affect the results, because it adds a vector proportional to e_i to every column of $\mathbf{B}$. Equation (9) shows that the columns of $\mathbf{B}$ are only involved in dot products with the eigenvectors of $\mathbf{A}$. It is easy to show that e_i is an eigenvector of $\mathbf{A}$ with eigenvalue zero. Therefore, all other eigenvectors must be orthogonal to e_i. Therefore, adding a vector proportional to e_i to the columns of $\mathbf{B}$ does not change the embedding results.

The computational complexity of LMDS is $O(Nmk + m^3)$. For large N, the computation time is dominated by computing the input distances from raw input data and projecting the distances in (10).

3 UNIFICATION OF THE THREE ALGORITHMS

While LMDS, FastMap, and MetricMap were all proposed independently and appear to be distinct algorithms, we will see in the following section that they are all, in fact, applications of the Nyström approximation.

3.1 FASTMAP

Consider LMDS for $k = 1$ and $m = 2$. In this case, we are trying to find a one-dimensional embedding using two landmark points. If the distance between the two landmarks is d_{12} and the distance from each landmark to all other points is d_{1i} or d_{2i}, then LMDS produces:

$$\mathbf{A} = \frac{1}{4}\begin{bmatrix} d_{12}^2 & -d_{12}^2 \\ -d_{12}^2 & d_{12}^2 \end{bmatrix}. \tag{16}$$

The largest eigenvalue for $\mathbf{A}$ is $d_{12}^2/2$, and its corresponding eigenvector is $[1\ -1]^T/\sqrt{2}$. After centering,

$$\mathbf{B} = \frac{1}{2}\begin{bmatrix} d_{1i}^2 - d_{12}^2/2 \\ d_{2i}^2 - d_{12}^2/2 \end{bmatrix}. \tag{17}$$

Using this in equation (10) yields a coordinate

$$x_i = \frac{\frac{-1}{2\sqrt{2}}\left(d_{1i}^2 - d_{2i}^2\right)}{d_{12}/\sqrt{2}} = \frac{d_{2i}^2 - d_{1i}^2}{2d_{12}}, \tag{18}$$

which is exactly one iteration of FastMap [7], except for an unimportant shift in the origin.

The $k = 1, m = 2$ case assumes that the principal eigenvector of the data lies on the line connecting points 1 and 2. This can be a poor estimate of this eigenvector. So, FastMap proposes a heuristic that scans a number of rows of the distance matrix $\mathbf{D}$, looking for two points that are far away from one another, assuming that the principal eigenvector is more likely to lie along the line connecting two points that are distant. Notice that if FastMap is going to generate m rows of a distance matrix per iteration, those rows can easily be added to the Nyström approximation to increase the accuracy of the eigenvalue and eigenvector estimate, instead of being used in a heuristic.

One iteration of FastMap generates only one embedding dimension at a time. Because FastMap is estimating the leading eigenvectors of a kernel matrix, it is legitimate to perform *deflation*. That is, each subsequent call to FastMap operates in a subspace that is orthogonal to previous dimensions, with previous eigenvectors projected away. Typically, deflation is performed by deducting a rank-one matrix from the kernel matrix. However, we cannot manipulate the entire kernel matrix. Therefore, FastMap computes the distances in the deflated space by deducting the squared distance between embedded coordinates from the original squared distance:

$$D_{ij}^2 = D_{i,j,\text{original}}^2 - \sum_n (x_{in} - x_{jn})^2. \qquad (19)$$

This is legitimate, because the deflated space is orthogonal to any embedded dimensions. Notice that this deflation can result in negative squared distances when the original distance matrix is not Euclidean. LMDS has a similar problem with negative eigenvalues of $\mathbf{A}$. FastMap produced negative distances for the Corel Features dataset in Section 4, while on the same data set, LMDS did not produce negative eigenvalues. These negative eigenvalues can be compensated by adding a small amount to the diagonal of $\mathbf{A}$.

The computational complexity of FastMap is $O(Nk^2)$, because the computation is dominated by k deflation iterations, each operating on N data samples, each of which takes $O(k)$ operations, due to (19).

3.2 METRICMAP

MetricMap [12] can be understood as a Nyström approximation by revisiting Landmark MDS. In LMDS, m rows of the distance matrix $\mathbf{D}$ are used to compute coordinates for every item. This rectangular slice is used both to find the coordinates of the landmarks (the m items corresponding to the m rows) at equation (8) and the remainder of the rows (equation (9)).

MetricMap uses a generalization of the Nyström approximation with different sized submatrices $\mathbf{A}$ and $\mathbf{B}$ in (8) and (9). MetricMap prescribes that $\mathbf{A}$ have dimension $2k \times 2k$, while $\mathbf{B}$ have dimension $k \times k$ [12]. Because the dimensions of $\mathbf{A}$ and $\mathbf{B}$ no longer match, the solution to the linear system in (9) is no longer correct. Instead, MetricMap derives embedding coordinates $\mathbf{X}$ from the $2k \times 2k$ matrix $\mathbf{A}$, using classical MDS from the k largest (in absolute value) eigenvalues of $\mathbf{A}$. Only k landmarks from the $2k$ embedded points are used:

$$\hat{\mathbf{X}} = \mathbf{\Gamma}_{[k,k]}^{1/2} \mathbf{U}_{[k,k]}, \qquad (20)$$

where the subscript $[k, k]$ indicates the sub-matrices indexed by the k largest eigenvalues and the k selected landmarks.

The k landmark rows of $\mathbf{A}$ that are selected by this procedure are also the rows used to generate $\mathbf{B}$. Thus, equation (9) becomes

$$\mathbf{Y} = \hat{\mathbf{X}}^{-T} \mathbf{B} = \mathbf{\Gamma}_{[k,k]}^{-1/2} \mathbf{U}_{[k,k]}^{-T} \mathbf{B}. \qquad (21)$$

The submatrix $\mathbf{U}_{[k,k]}$ must be inverted because it is no longer an orthogonal matrix. For speed, $\mathbf{U}_{[k,k]}$ undergoes LU decomposition, and each column of $\mathbf{B}$ is backsubstituted.

Note also that, as described in [12], MetricMap uses centering coefficients $c_i = \delta_{i1}$ (the first is one, the rest zero). This causes an unimportant shift in the origin of the coordinate system.

The computational complexity of MetricMap is $O(Nk^2 + k^3)$. In Landmark MDS, if m scales as k, then the computational complexity of MetricMap is the same as LMDS. LMDS may have an advantage in only requiring $O(k^2)$ work for every point in $\mathbf{B}$, rather than $O(km)$. However, because MetricMap uses different dimensions for $\mathbf{A}$ and $\mathbf{B}$, it is not known when MetricMap will yield an exact answer.

The unification of LMDS with MetricMap illustrates that the Nyström approximation can be generalized by allowing rows($\mathbf{A}$)$\geq$rows($\mathbf{B}$). Unlike the basic Nyström approximation, there are three free parameters: the final embedding dimension k, the number of rows in $\mathbf{B}$ and the number of rows in $\mathbf{A}$. The only required relationship between these is that rows($\mathbf{A}$)$\geq$rows($\mathbf{B}$)$\geq$ d. Thus, the restriction in [12] of rows($\mathbf{A}$)= 2 rows($\mathbf{B}$) is not required.

4 ACCURACY AND SPEED COMPARISONS

The main point of Section 3 is that FastMap, MetricMap, and LMDS are all applications of the Nyström

approximation. One important difference is that FastMap performs deflation, while MetricMap and LMDS require solving an eigensystem. Empirical testing can determine whether deflation is better than solving an eigensystem: does the eigensystem cause noticeable slowing? Does it yield extra accuracy?

Another difference is that MetricMap uses a matrix $\mathbf{A}$ that has more rows than $\mathbf{B}$, unlike FastMap and LMDS. Does using a larger matrix help the quality of the embedding?

To answer these questions, FastMap, MetricMap, and LMDS are compared on two medium-to-large data sets: the Reuters collection (a UCI KDD dataset [9]) and the Corel Image Features data set (also from UCI KDD).

4.1 REUTERS

The Reuters data set is a collection of Reuters news articles, stored in SGML. The algorithms are tested on the ModApte training set of Reuters, consisting of 9603 labeled documents, categorized with 115 labels, with multiple labels per document allowed.

The documents are converted into features vectors in a standard way. If the news article has an identified title and body, words in those are used. Otherwise, words in the text of the news article are used. All words are folded to lower case and then stemmed. Common words are removed from the list, and the remaining unique stemmed words in the corpus became features. The feature dimensionality is 22226, which is extremely high: larger than the number of examples in the set.

Each document i is represented by the standard tf-idf vector representation, denoted by t_{ik}, which is normalized to unit length. The distance squared matrix is then computed via

$$D_{ij}^2 = 1 - \sum_k t_{ik} t_{jk} \tag{22}$$

where the ith document is the ith row in t_{ik}.

FastMap, LMDS, and MetricMap are applied to the resulting distance matrix. The quality of the algorithms is measured in three ways: how much RMS relative distance error is introduced by the embedding, the F_1 retrieval quality score of the nearest neighbor in the embedded space, and CPU time. Two different settings of m are chosen for the LMDS experiments. First, $m = 3k$ is run. FastMap uses a heuristic to find the two farthest points that requires $3k$ distance rows to be computed. In order to match the computation of FastMap, LMDS is run with the same number of rows. The second experiment uses LMDS run with $m = 600$, which measures LMDS in a regime of high quality. MetricMap is run with rows($\mathbf{A}$) = 2 rows($\mathbf{B}$), as suggested by [12].

4.1.1 Relative Distance Error

The RMS relative distance error is measured by taking 100 documents at random from the set and measuring the 100×100 distance matrix, both in the original unembedded space, and in the embedded space, while varying the dimension k of the embedding. The RMS relative distance error is defined to be

$$\text{Error} = \sqrt{\frac{1}{10000} \sum_i (sE_i/t_i - 1)^2} \tag{23}$$

where t_i is the true (unembedded) distance, E_i is the estimated distance (in the embedded space), and s is a scaling factor. The lower this quantity, the better the embedding.

All three algorithms tend to underestimate the true distance between objects. However, in most applications, absolute distance is not needed: mean relative distance between items is the important quantity. A linear rescaling of the distance is thus harmless. Such a rescaling is reflected in s, which is the optimal scaling that maps E_i into t_i. This scaling is chosen to minimize the cost function in (23):

$$s = \frac{\sum_i E_i/t_i}{\sum_i E_i^2/t_i^2}. \tag{24}$$

Number of Dimensions	LMDS $m = 600$	LMDS $m = 3k$	FastMap	MetricMap
5	0.659	0.611	0.694	0.841
10	0.536	0.577	0.659	0.652
20	0.458	0.466	0.564	0.863
50	0.386	0.418	0.441	0.725
100	0.338	0.339	0.413	0.573
200	0.298	0.298	0.348	0.587

Table 1: RMS relative distance error on Reuters (lower is better).

The RMS error for Reuters is shown in Table 1. Three conclusions can be reached from this table.

First, when LMDS is only allowed to access $m = 3k$ distance rows (the same as FastMap), it still outperforms FastMap and MetricMap. This is because the heuristic in FastMap does not work well on text feature vectors: there are many documents that have very little overlap with each other, because their words do not overlap. The heuristic picks two examples that have little overlap as pivot points. The FastMap algorithm

then assigns most documents to the center of the coordinate, because most documents have little overlap with either of the two pivot points. Thus, it requires roughly twice as many dimensions to reach the same RMS error.

Second, further accuracy gains for LMDS can usually be had by increasing m above $3k$, although those gains are slight. These gains are shown in more detail, below.

Third, the extra information in the $2k \times 2k$ matrix $\mathbf{A}$ in MetricMap does not help the quality of the embedding: in fact, it actively harms it.

4.1.2 F_1 Retrieval Metric

Another metric for the algorithms is how the dimensionality reduction affects text retrieval and categorization. This is tested by finding, for each document, the nearest other document in the mapped space. Then, for all nearest neighbor pairs and for all labels, a standard microaveraged F_1 retrieval score is computed. Let A = the number of labels that are shared by any nearest pair. Let B = the number of labels that appear on one of the pair, but not the other. The F_1 score is then $F_1 = A/(A + B/2)$. The higher this quantity, the better the mapping is for text retrieval.

Number of Dimensions	LMDS $m = 600$	LMDS $m = 3k$	FastMap	MetricMap
5	0.520	0.458	0.467	0.396
10	0.625	0.581	0.521	0.412
20	0.714	0.653	0.629	0.489
50	0.754	0.717	0.709	0.430
100	0.768	0.757	0.724	0.434
200	0.768	0.768	0.753	0.346

Table 2: F_1 retrieval metric on Reuters (higher is better).

The F_1 metric for Reuters is shown in Table 2. There are several interesting results in this table. First, the F_1 retrieval metric when applied to the original unmapped distances is 0.719. Thus, above 100 dimensions, the F_1 score for the mapped distances can be better than the unmapped. These algorithms are implementing a form of Latent Semantic Analysis [6], which is known to improve retrieval. Second, the F_1 scores roughly track distance error: LMDS with $m = 600$ beats LMDS with $m = 3k$ beats FastMap beats MetricMap. Finally, for dimensions above 100, gains in F_1 performance start to asymptote.

Number of Dimensions	LMDS $m = 600$	LMDS $m = 3k$	FastMap	MetricMap
5	51.6	0.2	0.2	0.1
10	51.7	0.4	0.4	0.1
20	51.9	0.8	0.7	0.3
50	52.3	2.5	2.3	0.9
100	53.0	8.4	6.6	2.9
200	55.7	55.7	20.0	14.9

Table 3: CPU time (in seconds) on Reuters (lower is better).

4.1.3 CPU Time versus Accuracy

The CPU time for the three algorithms is shown in Table 3. The experiments are run on an unloaded 2.4 GHz Xeon PC running Windows Server 2003 with 1.5 GB of RAM. The datasets are small enough to fit into memory. Profiling the code shows that the time is dominated by the computation of the distance matrix elements: the eigendecomposition and projection take very little extra time.

Comparing FastMap and LMDS with $m = 3k$ is illustrative. For dimensions less than 200, the timings are very comparable. With $m = 3k$, LMDS does not incur a significant time penalty, but provides increased performance over FastMap. MetricMap is fastest, but the poor accuracy results in Table 2 indicate that the algorithm should be avoided.

Algorithm	m	RMS dist error	F_1	CPU time
FastMap	(150)	0.441	0.709	2.3
LMDS	60	0.424	0.687	0.9
LMDS	100	0.428	0.704	1.5
LMDS	150	0.417	0.717	2.5
LMDS	200	0.403	0.730	4.0
LMDS	300	0.380	0.742	8.1
LMDS	600	0.386	0.753	52.2

Table 4: Comparing performance of LMDS and FastMap on Reuters for different m ($k = 50$).

To better understand the behavior of LMDS, the tests are run for a fixed dimensionality $k = 50$ and for different values of m. The accuracy and CPU results are shown in Table 4. This table illustrates that LMDS permits a time/accuracy trade-off by varying m. Increasing m above $3k$ will increase the accuracy, at the cost of increased CPU. For this problem, the analysis time is not burdensome, so the increase accuracy is worthwhile. Eventually, the eigenvectors and eigenvalues of $\mathbf{K}$ are accurately estimated, so increasing m further does not help the accuracy.

4.2 COREL IMAGE FEATURES

The Corel Image Features is a UCI KDD data set consisting of features extracted from 68,040 images. Four sets of features are extracted: 32 color histograms, 32 color layout histograms, 9 color moments, and 16 texture features. Each of these features is continuous. No labels are provided in the data set. Images with missing features are ignored, leaving 66,615 images. The Corel Image Features dataset probes the algorithms in a different way. The data set size is larger and more realistic.

We computed an overall distance between two image feature vectors by first computing the distance between each set of features. For the color histograms and color layout histograms, we used the chi-squared distance [10]:

$$D_{ij}^{\text{chi}} = \sum_n \frac{2(h_{in} - h_{jn})^2}{h_{in} + h_{jn}} \tag{25}$$

where h_{in} is the nth histogram bin for ith image. For color moments and texture features, we used Euclidean distance (not squared).

At this point, there are four distances for each pair of images. These distances are combined into a single distance by a weighted sum, where the weights are computed so that the average of each of the four distances is unity across the database:

$$D_{ij}^{\text{total}} = \frac{D_{ij}^{\text{colorHist}}}{2.457} + \frac{D_{ij}^{\text{layout}}}{2.006} + \frac{D_{ij}^{\text{moments}}}{4.295} + \frac{D_{ij}^{\text{texture}}}{7.212}. \tag{26}$$

Again, the Corel Features dataset stresses the algorithms more than Reuters: the distance is not a Euclidean distance, but a sum of heterogeneous distances, which can provoke negative eigenvalues.

4.2.1 Relative Distance Error

Number of Dimensions	LMDS $m = 150$	LMDS $m = 3k$	FastMap	MetricMap
1	0.516	0.529	0.545	0.540
2	0.326	0.377	0.384	0.482
5	0.142	0.174	0.254	0.373
10	0.075	0.086	0.147	0.390
20	0.069	0.082	0.124	0.546
50	0.108	0.108	N/A	0.530

Table 5: RMS relative distance error on Corel Image Features (lower is better).

Because there are no image labels provided, we test the effectiveness of each algorithm with RMS relative distance error, as computed in (23). These accuracy results are shown in Table 5. There are severable notable features of the data in this table. First, FastMap yielded negative pivot distances for $k = 50$. That is, even the two farthest points in the database have negative squared distance after 50 rounds of deflation: no result for $k = 50$ could be produced by FastMap. Second, LMDS provides a noticeable improvement in accuracy for $k > 1$, even when $m = 3k$. LMDS with $m = 150$ is even more accurate. Third, as in Reuters, MetricMap has far worse accuracy than either of the other algorithms. Finally, the innate dimensionality of this data set may be less than 50, because the results for $k = 50$ are worse than for $k = 20$. This can happen, because the original distance matrix is non-Euclidean.

4.2.2 CPU Time versus Accuracy

Number of Dimensions	LMDS $m = 150$	LMDS $m = 3k$	FastMap	MetricMap
1	13.3	0.2	0.2	0.1
2	13.3	0.5	0.5	0.2
5	13.4	1.3	1.4	0.5
10	13.5	2.6	2.7	0.9
20	13.7	5.3	5.6	1.9
50	14.4	14.4	N/A	5.5

Table 6: CPU time (in seconds) on Corel Image Features (lower is better).

CPU times for the algorithms on this dataset are shown in Table 6. None of the CPU times are onerous. As in Reuters, LMDS for $m = 150$ is dominated by the computation of the distance matrix. However, for $m = 3k$, LMDS is competitive with FastMap. MetricMap is substantially faster (because it computes the fewest distance matrix elements), but its poor accuracy is not worth the increased speed.

Algorithm	m	RMS dist error	CPU time
FastMap	(60)	0.124	5.6
LMDS	40	0.125	3.6
LMDS	60	0.082	5.4
LMDS	100	0.072	9.0
LMDS	150	0.069	13.7
LMDS	200	0.069	18.7
LMDS	300	0.067	30.5

Table 7: Comparing performance of LMDS and FastMap on Corel Image Features for different m ($k = 20$).

Finally, the performance of LMDS is examined as a function of m for fixed $k = 30$ in Table 7. As m increases, LMDS surpasses the performance of FastMap until $m = 100$, where the accuracy is reaches a plateau and further increases in m do not help.

5 CONCLUSIONS

This paper has shown that FastMap, MetricMap, and Landmark MDS (LMDS) all utilize the Nyström approximation to the eigenvectors of a large matrix. This unification highlights how the algorithms are related and can lead to future work.

These algorithms use different variations on the Nyström approximation. LMDS uses the basic Nyström approximation. FastMap uses deflation to embed items one dimension at a time. MetricMap uses an expanded matrix to fix the embedding of the landmark points. However, empirical evidence has shown that the variations of the Nyström algorithm are not beneficial for MDS.

The deflation in FastMap limits the time/accuracy tradeoff that is inherent in the basic all-dimensions-at-once Nyström approximation. By increasing m (the number of rows computed in the original distance matrix), LMDS becomes more accurate, but the computation of the distance matrix elements require more time. FastMap freezes this tradeoff by choosing a constant number of rows per dimension. Even with the same number of rows of the distance matrix, LMDS is more accurate. Thus, the all-dimensions-at-once LMDS algorithm is preferred.

MetricMap uses a larger matrix than basic Nyström to compute the embedding coordinates of the landmark points. Empirically, this causes a substantial decrease in accuracy. This decrease in accuracy is probably because increasing the size of the $\mathbf{A}$ matrix does not improve the fitting of the landmark points: the number of landmark points grows as the dimension of $\mathbf{A}$. Thus, the extra data in $\mathbf{A}$ is not used to eliminate noise in the position on the landmark points.

In conclusion, the basic Nystroḿ approximation in LMDS is faster and more accurate than the Nystroḿ variations proposed in FastMap and MetricMap. This superiority may carry over to other data sets and applications other than MDS.

Acknowledgements

I would like to thank Chris Burges and Dimitris Achlioptas for discussions and help with the paper.

References

[1] S. Belongie, C. Fowlkes, F. Chung, and J. Malik. Spectral partitioning with indefinite kernels using the Nyström extension. In *Proc. ECCV*, 2002.

[2] Y. Bengio, J.-F. Paiement, and P. Vincent. Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps and spectral clustering. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Proc. NIPS*, volume 16, 2004.

[3] M. Chalmers. A linear iteration time layout algorithm for visualizing high-dimensional data. In *Proc. IEEE Information Visualization*, pages 127–132, 1996.

[4] T. Cox and M. Cox. *Multidimensional Scaling*. Number 59 in Monographs on Statistics and Applied Probability. Chapman & Hall, 1994.

[5] V. de Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In S. Becker, S. Thrun, and K. Obermayer, editors, *Proc. NIPS*, volume 15, pages 721–728, 2003.

[6] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[7] C. Faloutsos and K. Lin. FastMap: a fast algorithm for indexing, data-mining, and visualization. In *Proc. ACM SIGMOD*, pages 163–174, 1995.

[8] G. Golub and C. V. Loan. *Matrix Computations*. Johns Hopkins University Press, 1983.

[9] S. Hettich and S. Bay. The UCI KDD archive. [http://kdd.ics.uci.edu] Irvine, CA: UCI, Dept. Information and Computer Science.

[10] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw Hill, 1991.

[11] B. Schölkopf. The kernel trick for distances. In *Proc. NIPS*, pages 301–307, 2000.

[12] J. T.-L. Wang, X. Wang, K.-I. Lin, D. Shasha, B. A. Shapiro, and K. Zhang. Evaluating a class of distance-mapping algorithms for data mining and clustering. In *Proc. ACM KDD*, pages 307–311, 1999.

[13] C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, volume 13, pages 682–688, 2001.

[14] M. Williams and T. Munzner. Steerable, progressive multidimensional scaling. In *Proc. IEEE Information Visualization*, pages 57–64, 2004.

Bayesian Conditional Random Fields

Yuan (Alan) Qi
MIT CSAIL
32 Vassar Street
Cambridge, MA 02139
alanqi@csail.mit.edu

Martin Szummer
Microsoft Research
Cambridge, CB3 0FB
United Kingdom
szummer@microsoft.com

Thomas P. Minka
Microsoft Research
Cambridge, CB3 0FB
United Kingdom
minka@microsoft.com

Abstract

We propose Bayesian Conditional Random Fields (BCRFs) for classifying interdependent and structured data, such as sequences, images or webs. BCRFs are a Bayesian approach to training and inference with conditional random fields, which were previously trained by maximizing likelihood (ML) (Lafferty et al., 2001). Our framework eliminates the problem of overfitting, and offers the full advantages of a Bayesian treatment. Unlike the ML approach, we estimate the posterior distribution of the model parameters during training, and average over this posterior during inference. We apply an extension of EP method, the power EP method, to incorporate the partition function. For algorithmic stability and accuracy, we flatten the approximation structures to avoid two-level approximations. We demonstrate the superior prediction accuracy of BCRFs over conditional random fields trained with ML or MAP on synthetic and real datasets.

1 Introduction

Traditional classification models assume that data items are independent. However, real world data is often interdependent and has complex structure. Suppose we want to classify web pages into different categories, e.g., homepages of students versus faculty. The category of a web page is often related to the categories of pages linked to it. Rather than classifying pages independently, we should model them jointly to incorporate such contextual cues.

Joint modeling of structured data can be performed by generative graphical models, such as Bayesian networks or Markov random fields. For example, hidden Markov models have been used in natural language applications to assign labels to words in a sequence, where labels depend both on words and other labels along a chain. However, generative models have fundamental limitations. Firstly, generative models require specification of the data generation process, i.e., how data can be sampled from the model. In many applications, this process is unknown or impractical to write down, and not of interest for the classification task. Secondly, generative models typically assume conditional independence of observations given the labels. This independence assumption limits their modeling power and restricts what features can be extracted for classifying the observations. In particular, this assumption rules out features capturing long-range correlations, multiple scales, or other context.

Conditional random fields (CRF) are a conditional approach for classifying structured data, proposed by Lafferty et al. (2001). CRFs model only the label distribution conditioned on the observations. Unlike generative models, they do not need to explain the observations or features, and thereby conserve model capacity and reduce effort. This also allows CRFs to use flexible features such as complex functions of multiple observations. The modeling power of CRFs has shown great benefit in several applications, such as natural language parsing (Sha & Pereira, 2003), information extraction (McCallum, 2003), and image modeling (Kumar & Hebert, 2004).

To summarize, CRFs provide a compelling model for structured data. Consequently, there has been an intense search for effective training and inference algorithms. The first approaches maximized conditional likelihood (ML), either by generalized iterative scaling or by quasi-Newton methods (Lafferty et al., 2001; Sha & Pereira, 2003). However, the ML criterion is prone to overfitting the data, especially since CRFs are often trained with very large numbers of correlated features. The maximum a posteriori (MAP) criterion can reduce overfitting, but provides no guidance on the choice of parameter prior. Furthermore, large margin criteria have been applied to regularize the model parameters and also to kernelize CRFs (Taskar et al., 2004; Lafferty et al., 2004). Nevertheless, training and inference for CRFs remains a challenge, and the problems of overfitting, feature and model selection have largely remained open.

In this paper, we propose Bayesian Conditional Random Fields (BCRF), a novel Bayesian approach to training and inference for conditional random fields. Applying the Bayesian framework brings principled solutions and tools for addressing overfitting, model selection and many other aspects of the problem. Unlike ML, MAP, or large-margin approaches, we train BCRFs by estimating the posterior distribution of the model parameters. Subsequently, we can average over the posterior distribution for BCRF inference.

The complexity of the partition function in CRFs (the denominator of the likelihood function) necessitates approximations. Previous deterministic approximations including expectation propagation (EP) (Minka, 2001) or variational methods do not directly apply. In order to incorporate the partition function, we apply an extension of EP, the power EP method (Minka, 2004). Furthermore, we flatten the approximation structures for BCRFs to avoid two-level approximations. This significantly enhances the algorithmic stability and improves the estimation accuracy.

We first formally define CRFs, present the power EP method, and flatten the approximation structure for training. Then we propose an approximation method for model averaging, and finally show experimental results.

2 From conditional random fields to BCRFs

A conditional random field (CRF) models label variables according to an undirected graphical model conditioned on observed data (Lafferty et al., 2001). Let $\mathbf{x}$ be an "input" vector describing the observed data instance, and $\mathbf{t}$ be an "output" random vector over labels of the data components. We assume that all labels for the components belong to a finite label alphabet $\mathcal{T} = \{1, \ldots, T\}$. For example, the input $\mathbf{x}$ could be image features based on small patches, and $\mathbf{t}$ be labels denoting 'person, 'car' or 'other' patches. Formally, we have the following definition of CRFs (Lafferty et al., 2001):

Definition 2.1 *Let $G = (\mathcal{V}, \mathcal{E})$ be a graph such that $\mathbf{t}$ is indexed by the vertices of G. Then $(\mathbf{x}, \mathbf{t})$ is a conditional random field (CRF) if, when conditioned on $\mathbf{x}$, the random variables t_i obey the Markov property with respect to the graph: $p(t_i|\mathbf{x}, \mathbf{t}_{\mathcal{V}-i}) = p(t_i|\mathbf{x}, \mathbf{t}_{\mathcal{N}_i})$ where $\mathcal{V}-i$ is the set of all nodes in G except the node i, $\mathcal{N}_i$ is the set of neighbors of the node i in G, and $\mathbf{t}_\Omega$ represents the random variables of the vertices in the set Ω.*

Unlike traditional generative random fields, CRFs only model the conditional distribution $p(\mathbf{t}|\mathbf{x})$ and do not explicitly model the marginal $p(\mathbf{x})$. Note that the labels $\{t_i\}$ are globally conditioned on the whole observation $\mathbf{x}$ in CRFs. Thus, we do not assume that the observed data $\mathbf{x}$ are conditionally independent as in a generative random field.

BCRFs are a Bayesian approach to training and inference with conditional random fields. In some sense, BCRFs can be viewed as an extension of conditional Bayesian linear classifiers, e.g., Bayes point machines (BPM) (Herbrich et al., 1999; Minka, 2001), which are used to classify independent data points.

According to the Hammersley-Clifford theorem, a CRF defines the conditional distribution of the labels $\mathbf{t}$ given the observations $\mathbf{x}$ to be proportional to a product of potential functions on cliques of the graph G. For simplicity, we consider only pairwise clique potentials such that

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{w})} \prod_{\{i,j\}\in\mathcal{E}} g_{i,j}(t_i, t_j, \mathbf{x}; \mathbf{w}) \qquad (1)$$

where

$$Z(\mathbf{w}) = \sum_{\mathbf{t}} \prod_{\{i,j\}\in\mathcal{E}} g_{i,j}(t_i, t_j, \mathbf{x}; \mathbf{w}) \qquad (2)$$

is a normalizing factor known as the partition function, $g_{i,j}(t_i, t_j, \mathbf{x}; \mathbf{w})$ are pairwise potentials, and $\mathbf{w}$ are the model parameters. Note that the partition function is a complicated function of the model parameter $\mathbf{w}$. This makes Bayesian training much harder for CRFs than for Bayesian linear classifiers, since the normalizer of a Bayesian linear classifier is a constant.

In standard conditional random fields, the pairwise potentials are defined as

$$g_{i,j}(t_i, t_j, \mathbf{x}; \mathbf{w}) = \exp(\mathbf{w}_{t_i,t_j}^{\mathrm{T}} \boldsymbol{\phi}_{i,j}(\mathbf{x}, t_i, t_j)) \qquad (3)$$

where $\boldsymbol{\phi}_{i,j}(\mathbf{x}, t_i, t_j)$ are features extracted for the edge between vertices i and j of the conditional random field, and $\mathbf{w}_{t_i,t_j}$ are elements corresponding to labels $\{t_i, t_j\}$ in $\mathbf{w}$, where $\mathbf{w} = [\mathbf{w}_{1,1}^{\mathrm{T}}, \mathbf{w}_{1,2}^{\mathrm{T}}, \ldots, \mathbf{w}_{T,T}^{\mathrm{T}}]^{\mathrm{T}}$. There are no restrictions on the relation between features.

Instead of using an exponential potential function, we prefer to use the probit function $\Psi(\cdot)$ (the cumulative distribution function of a Gaussian with mean 0 and variance 1). This function is bounded, unlike the exponential, and permits efficient Bayesian training. Furthermore, to incorporate robustness against labeling errors, we allow a small probability ϵ of a label being incorrect, thus bounding the potential away from 0. Specifically, our robust potentials are:

$$g_{i,j}(t_i, t_j, \mathbf{x}; \mathbf{w}) = (1-\epsilon)\Psi(\mathbf{w}_{t_i,t_j}^{\mathrm{T}} \boldsymbol{\phi}_{i,j}(\mathbf{x}, t_i, t_j)) + \epsilon(1 - \Psi(\mathbf{w}_{t_i,t_j}^{\mathrm{T}} \boldsymbol{\phi}_{i,j}(\mathbf{x}, t_i, t_j))). \qquad (4)$$

Given the data likelihood and a Gaussian prior $p_0(\mathbf{w}) \sim \mathcal{N}(\mathbf{w}; \mathbf{0}, \mathrm{diag}(\boldsymbol{\alpha}))$, the posterior of the parameters is

$$p(\mathbf{w}|\mathbf{t}, \mathbf{x}) \propto \frac{1}{Z(\mathbf{w})} p_0(\mathbf{w}) \prod_{\{i,j\}\in\mathcal{E}} g_{i,j}(t_i, t_j, \mathbf{x}; \mathbf{w}) \qquad (5)$$

Expectation Propagation exploits the fact that the posterior is a product of simple terms. If we approximate each of these terms well, we can get a good approximation of the posterior. Mathematically, EP approximates $p(\mathbf{w}|\mathbf{t}, \mathbf{x})$ as

$$q(\mathbf{w}) = p_0(\mathbf{w}) \frac{1}{\tilde{Z}(\mathbf{w})} \prod_{\{i,j\} \in \mathcal{E}} \tilde{g}_{i,j}(\mathbf{w}) \tag{6}$$

$$= \frac{1}{\tilde{Z}(\mathbf{w})} \tilde{R}(\mathbf{w}) \tag{7}$$

where $\tilde{R}(\mathbf{w}) = p_0(\mathbf{w}) \prod_{\{i,j\} \in \mathcal{E}} \tilde{g}_{i,j}(\mathbf{w})$ is the numerator in $q(\mathbf{w})$. The approximation terms $\tilde{g}_{i,j}(\mathbf{w})$ and $\frac{1}{\tilde{Z}(\mathbf{w})}$ have the form of a Gaussian, so that the approximate posterior $q(\mathbf{w})$ is a Gaussian, i.e., $q(\mathbf{w}) \sim \mathcal{N}(\mathbf{m}_w, \mathbf{\Sigma}_w)$. We can approximate the pairwise potential functions $g_{i,j}(t_i, t_j, \mathbf{x}; \mathbf{w})$ by $\tilde{g}_{i,j}(\mathbf{w})$ in the numerator $\tilde{R}(\mathbf{w})$, just as in Bayesian linear classifiers (Minka, 2001). The main difficulty here is how to approximate the denominator $Z(\mathbf{w})$ by $\tilde{Z}(\mathbf{w})$ and incorporate it into $q(\mathbf{w})$.

3 EP and Power EP

This section reviews EP and presents power EP algorithms for BCRFs. Power EP is an extension of EP to make the computations more tractable. It was first used by Minka and Lafferty (2002), in the case of positive powers. However, power EP also works with negative powers, and this is one of the key insights that makes BCRF training tractable.

Given a distribution p written as a product of terms, and an approximating family q as in the previous section, Expectation Propagation tries to make q "close" to p in the sense of the Kullback Leibler divergence $KL(p||q) = \int p(\mathbf{w}) \log(p(\mathbf{w})/q(\mathbf{w})) d\mathbf{w}$. This is done by minimizing the divergence with respect to each term individually, holding the other terms fixed. We repeatedly cycle through all the terms until a fixed point is reached.

For the g_k terms, where k indexes edges, the algorithm first computes $q^{\backslash k}(\mathbf{w})$, which represents the "rest of the distribution." Then it minimizes KL-divergence over $\tilde{g}_k$, holding $q^{\backslash k}$ fixed. This process can be written succinctly as follows:

$$q^{\backslash k}(\mathbf{w}) \propto q(\mathbf{w}) / \tilde{g}_k(\mathbf{w}) \tag{8}$$

$$\tilde{g}_k(\mathbf{w})^{\text{new}} = \operatorname{argmin} KL(g_k(\mathbf{w}) q^{\backslash k}(\mathbf{w}) \;||\; \tilde{g}_k(\mathbf{w}) q^{\backslash k}(\mathbf{w})) \tag{9}$$

$$= \text{proj}\Big[g_k(\mathbf{w}) q^{\backslash k}(\mathbf{w}) \Big] / q^{\backslash k}(\mathbf{w}) \tag{10}$$

$$q(\mathbf{w})^{\text{new}} = q^{\backslash k}(\mathbf{w}) \tilde{g}_k(\mathbf{w})^{\text{new}} \tag{11}$$

Here proj is a "moment matching" operator: it finds the Gaussian having the same moments as its argument, thus minimizing KL. Algorithmically, (8) means "divide the Gaussians to get a new Gaussian, and call it $q^{\backslash k}(\mathbf{w})$." Similarly, (9) means "construct a Gaussian whose moments match $g_k(\mathbf{w}) q^{\backslash k}(\mathbf{w})$ and divide it by $q^{\backslash k}(\mathbf{w})$, to get a new Gaussian which replaces $\tilde{g}_k(\mathbf{w})$." This is the basic EP algorithm.

Power EP introduces a power n_k into EP and modifies the updates as follows:

$$q^{\backslash k}(\mathbf{w}) \propto q(\mathbf{w}) / \tilde{g}_k(\mathbf{w})^{1/n_k} \tag{12}$$

$$\tilde{g}_k(\mathbf{w})^{\text{new}} = \Big(\text{proj}\Big[g_k(\mathbf{w})^{1/n_k} q^{\backslash k}(\mathbf{w}) \Big] / q^{\backslash k}(\mathbf{w}) \Big)^{n_k} \tag{13}$$

$$q(\mathbf{w})^{\text{new}} = q(\mathbf{w}) \frac{\tilde{g}_k(\mathbf{w})^{\text{new}}}{\tilde{g}_k(\mathbf{w})} \tag{14}$$

As shown by Minka (2004), this update seeks to minimize a different measure of divergence, the α-divergence, where $\alpha = 2/n_k - 1$. Note that α can be any real number. By picking the power n_k appropriately, the updates can be greatly simplified. (This result is originally due to Wiegerinck and Heskes (2002). They discussed an algorithm called "fractional belief propagation" which is a special case of power EP, and all of their results also apply to power EP.)

We will use this update for the denominator term, so that g_k becomes $1/Z$, $\tilde{g}_k$ becomes $1/\tilde{Z}$, and pick $n_k = -1$:

$$q^{\backslash z}(\mathbf{w}) \propto q(\mathbf{w}) / \tilde{Z}(\mathbf{w}) = \tilde{R}(\mathbf{w}) / \tilde{Z}(\mathbf{w})^2 \tag{15}$$

$$\tilde{Z}(\mathbf{w})^{\text{new}} = \text{proj}\Big[Z(\mathbf{w}) q^{\backslash z}(\mathbf{w}) \Big] / q^{\backslash z}(\mathbf{w}) \tag{16}$$

$$q(\mathbf{w})^{\text{new}} = q(\mathbf{w}) \frac{\tilde{Z}(\mathbf{w})}{\tilde{Z}(\mathbf{w})^{\text{new}}} \tag{17}$$

In this way, we only need the moments of $Z(\mathbf{w})$, not $1/Z(\mathbf{w})$.

4 Approximating the partition function

In the moment matching step, we need to approximate the moments of $Z(\mathbf{w}) q^{\backslash z}(\mathbf{w})$. Murray and Ghahramani (2004) have proposed approximate MCMC methods to approximate the partition function of an undirected graph. This section presents an alternative method.

For clarity, let us rewrite the partition function as follows:

$$Z(\mathbf{w}) = \sum_{\mathbf{t}} Z(\mathbf{w}, \mathbf{t}) \tag{18}$$

$$Z(\mathbf{w}, \mathbf{t}) = \prod_{k \in \mathcal{E}} g_k(t_i, t_j, \mathbf{x}; \mathbf{w}) \tag{19}$$

where $k = \{i, j\}$ indexes edges. To compute the moments of $\mathbf{w}$, we can use EP recursively, to approximate $Z(\mathbf{w}, \mathbf{t}) q^{\backslash z}(\mathbf{w})$ as a function of $\mathbf{w}$ and $\mathbf{t}$. The approximation will have a factorized form:

$$q(\mathbf{w}) q(\mathbf{t}) = \tilde{Z}(\mathbf{w}) \tilde{Z}(\mathbf{t}) q^{\backslash z}(\mathbf{w}) \tag{20}$$

$$\tilde{Z}(\mathbf{w}) \tilde{Z}(\mathbf{t}) = \prod_{k \in \mathcal{E}} \tilde{f}_k(\mathbf{w}) \tilde{f}_k(t_i) \tilde{f}_k(t_j) \tag{21}$$

where $\tilde{f}_k(\mathbf{w})\tilde{f}_k(t_i)\tilde{f}_k(t_j)$ approximates $g_k(t_i, t_j, \mathbf{x}; \mathbf{w})$ in the denominator. Note that this $q(\mathbf{w})$ is the same as the overall $q(\mathbf{w})$ at the convergence.

Because the approximation is factorized, the computation will have the flavor of loopy belief propagation. The initial $\tilde{f}_k$ will be 1, making the initial $q(\mathbf{w}) = q^{\backslash z}(\mathbf{w})$ and $q(\mathbf{t}) = 1$. The EP update for $\tilde{f}_k$ is:

$$q^{\backslash k}(\mathbf{w}) \propto q(\mathbf{w})/\tilde{f}_k(\mathbf{w}) \tag{22}$$

$$q^{\backslash k}(t_i) \propto q(t_i)/\tilde{f}_k(t_i) \qquad \text{(similarly for } j\text{)} \tag{23}$$

$$f_k(\mathbf{w}) = \sum_{t_i,t_j} g_k(t_i, t_j, \mathbf{x}; \mathbf{w}) q^{\backslash k}(t_i) q^{\backslash k}(t_j) \tag{24}$$

$$\tilde{f}_k(\mathbf{w})^{\text{new}} = \text{proj}\Big[q^{\backslash k}(\mathbf{w}) f_k(\mathbf{w})\Big] / q^{\backslash k}(\mathbf{w}) \tag{25}$$

$$\tilde{f}_k(t_i)^{\text{new}} = \sum_{t_j} \int_{\mathbf{w}} g_k(t_i, t_j, \mathbf{x}; \mathbf{w}) q^{\backslash k}(\mathbf{w}) q^{\backslash k}(t_j) d\mathbf{w} \tag{26}$$

$$q(\mathbf{w})^{\text{new}} = q^{\backslash k}(\mathbf{w}) \tilde{f}_k(\mathbf{w})^{\text{new}} \tag{27}$$

$$q(t_i)^{\text{new}} = q^{\backslash k}(t_i) \tilde{f}_k(t_i)^{\text{new}} \tag{28}$$

These updates are iterated for all k, until a fixed point is reached. A straightforward implementation of the above updates costs $O(d^3)$ time, where d is the dimension of the parameter vector $\mathbf{w}$, since they involve inverting the covariance matrix of $\mathbf{w}$. However, as shown in the next section, it is possible to do them with low-rank matrix updates, in $O(d^2)$ time.

5 Efficient low-rank matrix computation for moments

First, let us define $\boldsymbol{\phi}_k(m, n, \mathbf{x})$ as shorthand of $\boldsymbol{\phi}_k(t_i = m, t_j = n, \mathbf{x})$, where $\boldsymbol{\phi}_k(t_i, t_j, \mathbf{x})$ are feature vectors extracted at edge $k = \{i, j\}$ with labels t_i and t_j on nodes i and j, respectively. Then we have

$$\mathbf{A}_k = \begin{pmatrix} \boldsymbol{\phi}_k(1,1,\mathbf{x}) & 0 & \cdots & 0 \\ 0 & \boldsymbol{\phi}_k(1,2,\mathbf{x}) & 0 & \cdots \\ 0 & \cdots & 0 & \boldsymbol{\phi}_k(T,T,\mathbf{x}) \end{pmatrix}$$

$$\mathbf{y} = \mathbf{A}_k^{\mathrm{T}} \mathbf{w} \tag{29}$$

$$f_k(\mathbf{y}) = \sum_{t_i,t_j} \Psi(y_{t_i,t_j}) q^{\backslash k}(t_i) q^{\backslash k}(t_j) \tag{30}$$

where $y_{t_i,t_j} = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}_k(t_i, t_j, \mathbf{x})$. Clearly, we can rewrite $q(\mathbf{w})$ as $f_k(\mathbf{y}) q^{\backslash k}(\mathbf{w})$. Since the dimensionality of $\mathbf{y}$ is usually a lot smaller than that of $\mathbf{w}$, the exact term $f_k(\mathbf{y})$ only constrains the distribution $q(\mathbf{w})$ in a smaller subspace. Therefore, it is sensible to use low-rank matrix computation to obtain $\mathbf{m}_w$ and $\mathbf{V}_w$, the mean and variance of $q(\mathbf{w})$.

The derivation is omitted because of the space limitation. The details can be found in Qi (2004). Here, we simply give the updates:

$$\mathbf{m}_w = \mathbf{m}_w^{\backslash k} + \mathbf{V}_w^{\backslash k} \mathbf{A}_k \mathbf{c} \tag{31}$$

$$\mathbf{V}_w = \mathbf{V}_w^{\backslash k} - \mathbf{V}_w^{\backslash k} \mathbf{A}_k \mathbf{D} \mathbf{A}_k^{\mathrm{T}} \mathbf{V}_w^{\backslash k} \tag{32}$$

$$\mathbf{c} = (\mathbf{V}_y^{\backslash k})^{-1} (\mathbf{m}_y - \mathbf{m}_y^{\backslash k}) \tag{33}$$

$$\mathbf{D} = (\mathbf{V}_y^{\backslash k})^{-1} - (\mathbf{V}_y^{\backslash k})^{-1} (\mathbf{G}_y - \mathbf{m}_y \mathbf{m}_y^{\mathrm{T}}) (\mathbf{V}_y^{\backslash k})^{-1} \tag{34}$$

where

$$Z = \sum_{t_i,t_j} q^{\backslash k}(t_i, t_j) \int \Psi(y_{t_i,t_j}) \mathcal{N}(\mathbf{y} | \mathbf{m}_y^{\backslash k}, \mathbf{V}_y^{\backslash k}) \mathrm{d}\mathbf{y} \tag{35}$$

$$= \sum_{t_i,t_j} q^{\backslash k}(t_i) q^{\backslash k}(t_j) Z_{t_i,t_j} \tag{36}$$

$$\mathbf{m}_y = \frac{\int f_k(\mathbf{y}) \mathbf{y} \mathcal{N}(\mathbf{y} | \mathbf{m}_y^{\backslash k}, \mathbf{V}_y^{\backslash k})}{Z} \tag{37}$$

$$= \frac{\sum_{t_i,t_j} Z_{t_i,t_j} \mathbf{m}_{y_{t_i,t_j}} q^{\backslash k}(t_i) q^{\backslash k}(t_j)}{Z} \tag{38}$$

$$\mathbf{G}_y = \frac{\int f_k(\mathbf{y}) \mathbf{y}\mathbf{y}^{\mathrm{T}} \mathcal{N}(\mathbf{y} | \mathbf{m}_y^{\backslash k}, \mathbf{V}_y^{\backslash k})}{Z} \tag{39}$$

$$= \frac{\sum_{t_i,t_j} Z_{t_i,t_j} \mathbf{G}_{y_{t_i,t_j}} q^{\backslash k}(t_i) q^{\backslash k}(t_j)}{Z} \tag{40}$$

where

$$z_{t_i,t_j} = \frac{\mathbf{e}_k \mathbf{m}_y^{\backslash k}}{\sqrt{\mathbf{e}_k \mathbf{V}_y^{\backslash k} \mathbf{e}_k^{\mathrm{T}} + 1}} \tag{41}$$

$$\rho_{t_i,t_j} = \frac{1}{\sqrt{\mathbf{e}_k \mathbf{V}_y^{\backslash k} \mathbf{e}_k^{\mathrm{T}} + 1}} \frac{(1 - 2\epsilon \mathcal{N}(z_{t_i,t_j} | 0, 1))}{\epsilon + (1 - 2\epsilon) \Psi(z_{t_i,t_j})} \tag{42}$$

$$Z_{t_i,t_j} = \int \Psi(y_{i,j}) \mathcal{N}(\mathbf{y} | \mathbf{m}_y^{\backslash k}, \mathbf{V}_y^{\backslash k}) \mathrm{d}\mathbf{y} \tag{43}$$

$$= \int \Psi(\mathbf{e}_k \mathbf{y}) \mathcal{N}(\mathbf{y} | \mathbf{m}_y^{\backslash k}, \mathbf{V}_y^{\backslash k}) \mathrm{d}\mathbf{y} \tag{44}$$

$$= \epsilon + (1 - 2\epsilon) \Psi(z_{t_i,t_j}) \tag{45}$$

$$\mathbf{m}_{y_{t_i,t_j}} = \frac{\int \Psi(\mathbf{e}_k \mathbf{y}) \mathbf{y} \mathcal{N}(\mathbf{y} | \mathbf{m}_y^{\backslash k}, \mathbf{V}_y^{\backslash k}) \mathrm{d}\mathbf{y}}{Z_{t_i,t_j}} \tag{46}$$

$$= \mathbf{m}_y^{\backslash k} + \mathbf{V}_y^{\backslash k} \rho_{t_i,t_j} \mathbf{e}_k^{\mathrm{T}} \tag{47}$$

$$\mathbf{G}_{y_{t_i,t_j}} = \mathbf{V}_y^{\backslash k} - \mathbf{V}_y^{\backslash k} \mathbf{e}_k^{\mathrm{T}} \Big(\frac{\rho_{t_i,t_j} (\mathbf{e}_k \mathbf{m}_{y_{t_i,t_j}} + \rho_{t_i,t_j})}{\mathbf{e}_k \mathbf{V}_y^{\backslash k} \mathbf{e}_k' + 1} \Big) \mathbf{e}_k \mathbf{V}_y^{\backslash k} \tag{48}$$

where $\mathbf{e}_k$ is a vector with all elements being zeros except its k^{th} element being one.

We update $q(t_i)$ and $q(t_j)$ as follows:

$$q(t_i, t_j) = \frac{Z_{t_i,t_j} q^{\backslash k}(t_i) q^{\backslash k}(t_j)}{Z} \tag{49}$$

$$q(t_i) = \sum_{t_j} q(t_i, t_j), \quad q(t_j) = \sum_{t_i} q(t_i, t_j) \tag{50}$$

6 Flattening the approximation structure

In practice, we found that the approximation method, presented in Sections 3 and 4, led to non-positive covariance matrices in training. In this section, we examine the reason for this problem and propose a method to fix it.

The approximation method has two levels of approximations, which are visualized at the top of Figure 1. At the upper level of the top graph, the approximation method iteratively refines the approximate posterior $q(\mathbf{w})$ based on the term approximation $\tilde{Z}(\mathbf{w})$; at the lower level, it iteratively refines $\tilde{Z}(\mathbf{w})$ by smaller approximation terms $\{\tilde{f}_k(\mathbf{w})\}$.

A naive implementation of the two-level approximation will always initialize the approximation terms $\{\tilde{f}_k(\mathbf{w})\}$ as 1, such that the iterations at the lower level start from scratch every time. Thus, removing the denominator $\tilde{Z}(\mathbf{w})$ in the upper level amounts to removing all the previous approximation terms $\{\tilde{f}_k(\mathbf{w})\}$ in the lower level. The naive implementation requires the "leave-one-out" approximation $q^{\setminus z}(\mathbf{w}) \propto \frac{q(\mathbf{w})}{\tilde{Z}(\mathbf{w})}$ to have a positive definite covariance matrix. Since this requirement is hard to meet, the training procedure often skips the whole denominator $Z(\mathbf{w})$ in the upper level. This skipping would dramatically decrease the approximation accuracy in practice.

A better idea is to keep the values of the approximation terms and initialize the approximation terms using the values obtained from the previous iterations. By doing so, we do not require the covariance of $q^{\setminus z}(\mathbf{w})$ to be positive definite anymore. Instead, we need that of $q^{\setminus k}(\mathbf{w})$ in equation (22) to be positive definite, which is easier to satisfy. Now, when the iterations in the lower level start, $q^{\setminus k}(\mathbf{w})$ usually has a positive definite covariance. However, after a few iterations, the covariance of $q^{\setminus k}(\mathbf{w})$ often becomes non-positive definite again. The underlying reason is that the partition function $Z(\mathbf{w})$ is a complicated function, which is difficult to be accurately approximated by EP.

To address the problem, we flatten the two-level approximation structure by expanding $\tilde{Z}(\mathbf{w})$ in the upper level. Now we focus on $q(\mathbf{w})$, which is of our interest in training, without directly approximating the difficult partition function $Z(\mathbf{w})$ in the intermediate step. The flattened structure is shown at the bottom of the Figure 1. Specifically, the approximate posterior $q(\mathbf{w})$ has the following form:

$$q(\mathbf{w}) \propto p_0(\mathbf{w}) \prod_{k \in \mathcal{E}} \tilde{g}_k(\mathbf{w}) \frac{1}{\prod_{k \in \mathcal{E}} \tilde{f}_k(\mathbf{w})} \tag{51}$$

Equation (51) uses the approximation term $\tilde{f}_k(\mathbf{w})$ for each edge rather than using $\tilde{Z}(\mathbf{w})$. It is also possible to interpret the flattened structure from the perspective of the two-level approximation structure. That is, each time we partially update $\tilde{Z}(\mathbf{w})$ based on only one small term approximation $\tilde{f}_k(\mathbf{w})$, and then refine $q(\mathbf{w})$ before updating $\tilde{Z}(\mathbf{w})$ again based on another small term approximation.

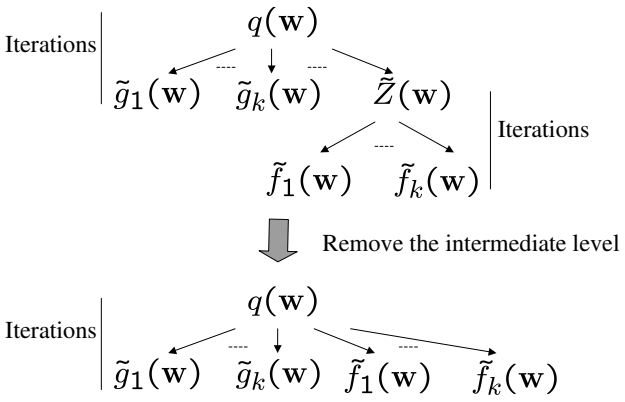


Figure 1: Flattening the approximation structure. The upper graph shows the two-level approximation structure of the methods described in the previous sections. The lower graph shows the flattened single-level approximation structure.

With the flattened structure, the deletion steps for removing $\tilde{g}_k(\mathbf{w})$ and $\tilde{f}_k(\mathbf{w})$ remain the same as before. The moment matching steps are the same too. We only need to assign negative power one to the approximation terms in the denominators. Specifically, we have the following updates:

$$\mathbf{h}_k = \left(\mathbf{D}^{-1} - \mathbf{A}_k^{\mathrm{T}} \mathbf{V}_w^{\setminus k} \mathbf{A}_k\right)^{-1} \tag{52}$$

$$\boldsymbol{\mu}_k = \mathbf{c} + \mathbf{h}_k \mathbf{A}_k^{\mathrm{T}} \mathbf{m}_w \tag{53}$$

$$\boldsymbol{\xi}_k = 2\mathbf{h}_k^{old} - \mathbf{h}_k \quad \boldsymbol{\eta}_k = 2\boldsymbol{\mu}_k^{old} - \boldsymbol{\mu}_k \tag{54}$$

$$\mathbf{V}_w = \mathbf{V}_w^{\setminus k} - \mathbf{V}_w^{\setminus k} \mathbf{A}_k (\boldsymbol{\xi}_k^{-1} + \mathbf{A}_k^{\mathrm{T}} \mathbf{V}_w^{\setminus k} \mathbf{A}_k)^{-1} \mathbf{A}_k^{\mathrm{T}} \mathbf{V}_w^{\setminus k} \tag{55}$$

$$\mathbf{m}_w = \mathbf{m}_w^{\setminus k} - \mathbf{V}_w^{\setminus k} \mathbf{A}_k (\boldsymbol{\xi}_k^{-1} + \mathbf{A}_k^{\mathrm{T}} \mathbf{V}_w^{\setminus k} \mathbf{A}_k)^{-1} \mathbf{A}_k^{\mathrm{T}} \mathbf{m}_w^{\setminus k} + \mathbf{V}_w \mathbf{A}_k \boldsymbol{\eta}_k \tag{56}$$

where $\mathbf{c}$ and $\mathbf{D}$ are defined in Equations (33) and (34). Note that the above computation takes $O(d^2)$ time, while a simple implementation of the two-level approximation would take $O(d^3)$ time due to the inverse of the covariance matrix of $\tilde{Z}(\mathbf{w})$.

As a result of structure flattening, we have the following advantages over the previous two approaches. First, in our experiments, training can converge easily. Instead of iteratively approximating $Z(\mathbf{w})$ in the lower level based all the small terms as before, a partial update based on only one small term makes training much more stable. Second, we can obtain a better "leave-one-out" posterior $q^{\setminus k}(\mathbf{w})$, since we update $q(\mathbf{w})$ more frequently than in the previous two cases. A more refined $q(\mathbf{w})$ leads to a better $q^{\setminus k}(\mathbf{w})$, which in turn guides the KL mininmization to find a better new $q(\mathbf{w})$. Finally, the flattened approximation structure allows us to process approximation terms in any order. We found empirically that, compared to using a random order, it is better to process the denominator term $\{\tilde{f}_k(\mathbf{w})\}$ right after processing the numerator term $\{\tilde{g}_k(\mathbf{w})\}$, which is associ-

ated with the same edge as $\{\tilde{f}_k(\mathbf{w})\}$.

Using the flattened structure and pairing the processing of the corresponding numerator and denominator terms, we can train BCRFs robustly. For example, on the tasks of analyzing synthetic datasets in the experimental section, training with the two-level structure breaks down by skipping $\{\tilde{Z}(\mathbf{w})\}$ or $\{\tilde{f}_k(\mathbf{w})\}$ and fails to converge. In contrast, training with the flattened structure converges successfully and leads to a test error around 10%.

7 Inference by approximate model averaging

Unlike traditional classification problems, where we have a scalar output for each input, a BCRF jointly labels all the hidden vertices in an undirected graph. The trained BCRF infer the lables by model averaging, which makes full use of the training data by employing not only the estimated mean of the parameters, but also the estimated uncertainty (variance).

Given a new graph $\mathbf{x}^\star$, a BCRF trained on $(\mathbf{x}, \mathbf{t})$ can approximate the predictive distribution as follows:

$$p(\mathbf{t}^\star|\mathbf{x}^\star, \mathbf{t}, \mathbf{x}) = \int p(\mathbf{t}^\star|\mathbf{x}^\star, \mathbf{w}) p(\mathbf{w}|\mathbf{t}, \mathbf{x}) d\mathbf{w} \tag{57}$$

$$\approx \int p(\mathbf{t}^\star|\mathbf{x}^\star, \mathbf{w}) q(\mathbf{w}) d\mathbf{w} \tag{58}$$

$$= \int \frac{q(\mathbf{w})}{Z(\mathbf{w})} \prod_{\{i,j\}\in\mathcal{E}} g_{i,j}(t_i^\star, t_j^\star, \mathbf{x}^\star; \mathbf{w}) d\mathbf{w}$$

where $q(\mathbf{w})$ is the approximation of the true posterior $p(\mathbf{w}|\mathbf{t}, \mathbf{x})$. Since the exact integration for model averaging is intractable, we need to approximate this integral.

We can approximate the predictive posterior term by term as in EP. But typical EP updates will involve the updates over both the parameters and the labels. That is much more expensive than using a point estimate for inference, which invloves only updates of the labels. To reduce the computational complexity, we propose the following approach. It is based on the simple fact that without any label, the test data point does not offer information about the parameters in the conditional model, since we do not couple BCRFs with semi-supervised learning. Since $q(\mathbf{w})$ is unchanged, we can only update $q(\mathbf{t}^\star)$ when incorporating one term $g_{i,j}(t_i^\star, t_j^\star, \mathbf{x}; \mathbf{w})$. Specifically, given the posterior $q(\mathbf{w}) \sim \mathcal{N}(\mathbf{m}_w, \mathbf{V}_w)$, we use the factorized approximation $q(\mathbf{t}^\star) = \prod_i q(t_i^\star)$ and update $q(t_i^\star)$ and $q(t_j^\star)$ as follows:

$$z_{t_i^\star, t_j^\star} = \frac{\boldsymbol{\phi}_k^{\mathrm{T}} \mathbf{m}_w^{\backslash k}}{\sqrt{\boldsymbol{\phi}_k^{\mathrm{T}} \mathbf{V}_w^{\backslash k} \boldsymbol{\phi}_k + 1}} \tag{59}$$

$$Z_{t_i^\star, t_j^\star} = \epsilon + (1 - 2\epsilon)\Psi(z_{t_i^\star, t_j^\star}) \tag{60}$$

$$q(t_i^\star, t_j^\star) = \frac{Z_{t_i^\star, t_j^\star} q^{\backslash k}(t_i^\star) q^{\backslash k}(t_j^\star)}{Z} \tag{61}$$

$$q(t_i^\star) = \sum_{t_j} q(t_i^\star, t_j^\star) \tag{62}$$

$$q(t_j^\star) = \sum_{t_i} q(t_i^\star, t_j^\star) \tag{63}$$

where $\boldsymbol{\phi}_k$ is the feature vector extracted at the k^{th} edge. Note that the deletion and inclusion steps for $q(\mathbf{t}^\star)$ are similar to those in BCRF training.

8 Experimental results

This section compares BCRFs with CRFs trained by maximum likelihood (ML) and maximum a posteriori (MAP) methods on several synthetic datasets and a document labeling task, demonstrating BCRFs' superior test performance. MAP-trained CRFs include CRFs with probit potential functions (4) and CRFs with exponential potential functions (3). We used probit models as potential functions by setting $\epsilon = 0$ in equation (4). In BCRF training, we used a small step size to avoid divergence (see details in Qi (2004)). For comparsion, the errors were counted on all vertices in the test graphs.

8.1 Synthetic CRFs classification

All the synthetic datasets were sampled from CRFs with probit potential functions (4). On these synthetic datasets, we compared the test performance of BCRFs and MAP-trained probit CRFs for different sizes of training sets and different graphical structures.

The labels of the vertices in synthetic graphs are all binary. The parameter vector $\mathbf{w}$ has 24 elements. The feature vectors $\{\boldsymbol{\phi}_{i,j}\}$ are randomly sampled from one of four Gaussians. We can easily control the discriminability of the data by changing the variance of the Gaussians. Based on the model parameter vector and the sampled feature vectors, we can compute the joint probability of the labels as in equation (1) and randomly sample the labels. For BCRFs, we used a step size of 0.8 for training. For MAP-trained CRFs, we used quasi-Newton methods with the BFGS approximation of Hessians (Sha & Pereira, 2003).

8.1.1 Different training sizes for loopy CRFs

Each graph has 3 vertices in a loop. In each trial, 10 loops were sampled for training and 1000 loops for testing. The procedure was repeated for 10 trials. A Gaussian prior with mean $\mathbf{0}$ and diagonal variance $5\mathbf{I}$ was used for both BCRF and MAP CRF training. For ML- and MAP-trained CRFs, we applied the junction tree algorithm for inference. For BCRFs, we used approximate model averaging for inference. We repeated the same experiments by increasing the number of training graphs from 10 to 30 to 100.

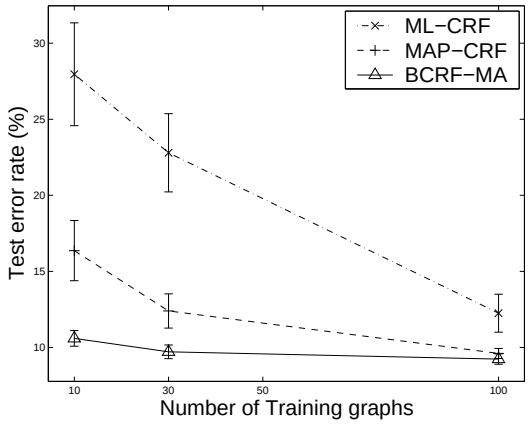


Figure 2: Test error rates for MAP-trained CRFs and BCRFs on synthetic datasets with different numbers of training loops. The results are averaged over 10 runs. Each run has 1000 test loops. Non-overlapping of error bars, the standard errors scaled by 1.64, indicates 95% significance of the performance difference.

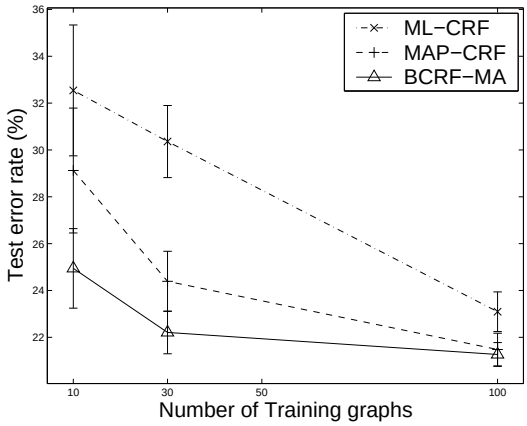


Figure 3: Test error rates for MAP-trained CRFs and BCRFs on synthetic datasets with different numbers of training chains. The results are averaged over 10 runs. Each run has 1000 test chains. Though the error bars overlap a little bit, BCRFs still outperform ML- and MAP-trained CRFs at 95% significance according to t-tests.

The results are visualized in Figure 2. According to t-tests, which have stronger test power than the error bars in Figure 2, BCRFs outperform ML- and MAP-trained CRFs in all cases at 98% statistical significance level. When more training graphs are available, MAP-trained CRFs and BCRFs perform increasingly similarly, though still statistically differently. The reason is that the posterior is narrower than in the case of fewer training graphs, such that the posterior mode is closer to the posterior mean.

8.1.2 Different training sizes for chain-structured CRFs

We then changed the graphs to be chain-structured. Specifically, each graph has 3 vertices in a chain. In each trial, 10 chains were sampled for training and 1000 chains for testing. The procedure was repeated for 10 trials. A Gaussian prior with mean $\mathbf{0}$ and diagonal variance $5\mathbf{I}$ was used for both BCRF and MAP CRF training. Then we repeated the same experiments by increasing the number of training graphs from 10 to 30 to 100. The results are visualized in Figure 3. Again, BCRFs outperform ML- and MAP-trained CRFs with high statistical significance.

8.2 FAQ labeling

We compared BCRFs with MAP-trained probit and exponential CRFs on the frequently asked questions (FAQ) dataset, introduced by McCallum et al. (2000). The dataset consists of 47 files, belonging to 7 Usenet newsgroup FAQs. Each files has multiple lines, which can be the header (H), a question (Q), an answer (A), or the tail (T). Since identifying the header and the tail is relatively easy, we simplify the task to label only the lines that are questions or answers. To save time, we truncated all the FAQ files such that no file has more than 500 lines. On average, the truncated files have 479 lines. The dataset was randomly split 10 times into 19 training and 28 test files. Each file was modeled by a chain-structured CRF, whose vertices correspond to lines in the files. The feature vector for each edge of a CRF is simply the concatenation of feature vectors extracted at the two neighboring vertices.

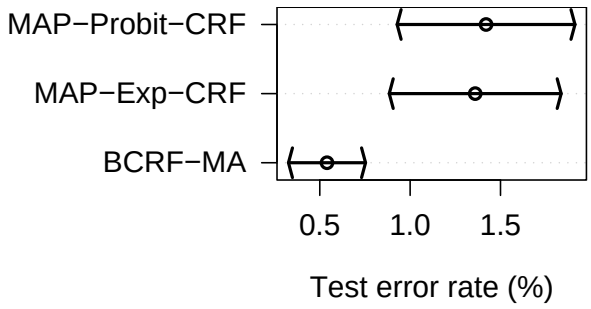


Figure 4: Test error rates of different algorithms on FAQ dataset. The results are averaged over 10 random splits. Non-overlapping of the error bars, the standard errors multiplied by 1.64, indicates that BCRFs outperform MAP-trained CRFs with probit and exponential potentials at 95% statistical significance level.

The test performance is visualized in Figure 4. According to t-tests, BCRFs outperform ML- and MAP-trained CRFs with probit or exponential potentials on the truncated FAQ dataset at 98% statistical significance level.

8.3 Comparing computational complexity

In general, the computational cost of ML and Bayesian training depends on many factors. On the one hand, for ML and MAP training, the cost of the BFGS algorithm is $O(d \max\{d, |\mathcal{E}|\})$ per iteration, where d is the length of $\mathbf{w}$, and $|\mathcal{E}|$ is the total number of edges in training graphs. The cost of BCRF training is $O(|\mathcal{E}|d^2)$ per iteration. Therefore BCRF training is about as $\min\{d, |\mathcal{E}|\}$ times expensive as ML and MAP training per iteration. On the other hand, BCRF training generally takes much fewer iterations to converge than BFGS training. Moreover, in BFGS training there is an embedded inference problem to obtain the needed statistics for optimization. This inference problem can be relatively expensive and cause a big hidden constant in $O(d \max\{d, |\mathcal{E}|\})$, the BFGS cost per iteration. On synthetic data where the number of edges is limited, BCRF training is at least as efficient as BFGS training. For example, given the 10 training sets in Section 8.1.2, each of which has 30 chain-structured CRFs, BCRF and BFGS training on a Pentium 4 3.1GHz computer used 8.81 and 21.16 seconds on the average, respectively. On real-world data, many factors play together to determine the efficiency of a training method. On a random split of the FAQ dataset where the total number of edges in graphs is more than 8000, it took about 9 and 2 hours (443 and 130 iterations) for BFGS to train probit and exponential CRFs, while it took BCRF training about 6 hours (30 iterations).

9 Conclusions

This paper has presented BCRFs, a new approach to training and inference on conditional random fields. In training, BCRFs approximate the posterior distribution of the parameters using a variant of the power EP method. Also, BCRFs flatten approximation structures to increase the algorithmic stability, efficiency, and prediction accuracy. In testing, BCRFs use approximate model averaging. On synthetic data and FAQ files, we compared BCRFs with ML- and MAP-trained CRFs. In almost all the experiments, BCRFs outperformed ML- and MAP-trained CRFs significantly.

Compared to ML- and MAP-trained CRFs, BCRFs can approximate model averaging over the posterior distribution of the parameters, instead of using a MAP or ML point estimate of the parameter vector for inference. Furthermore, BCRF hyperparameters can be optimized in a principled way, such as by maximizing the evidence, with parameters integrated out. EP returns an estimate of the evidence as a by-product. Similarly, we can use the method by Qi et al. (2004) to do feature selection with BCRFs and to obtain sparse kernelized BCRFs. More importantly, the techniques developed for BCRFs have promise for Bayesian learning in Markov networks.

Acknowledgment

Y. Qi was supported by the Things That Think consortium at the MIT Media laboratory before he moved to CSAIL. Thanks to Andrew McCallum for providing the features used in the FAQ dataset and Fei Sha for offering the maximum likelihood training code for CRFs with exponential potentials.

References

Herbrich, R., Graepel, T., & Campbell, C. (1999). Bayes point machine: Estimating the Bayes point in kernel space. *IJCAI Workshop SVMs* (pp. 23–27).

Kumar, S., & Hebert, M. (2004). Discriminative fields for modeling spatial dependencies in natural images. *Advances in Neural Information Processing Systems 16*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proc. 18th International Conf. on Machine Learning* (pp. 282–289). Morgan Kaufmann, San Francisco, CA.

Lafferty, J., Zhu, X., & Liu, Y. (2004). Kernel conditional random fields: Representation and clique selection. *Proc. International Conf. on Machine Learning*.

McCallum, A. (2003). Efficiently inducing features of conditional random fields. *Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence*.

McCallum, A., Freitag, D., & Pereira, F. (2000). Maximum entropy Markov models for information extraction and segmentation. *Proc. 17th International Conf. on Machine Learning* (pp. 591–598). Morgan Kaufmann, San Francisco, CA.

Minka, T. P. (2001). Expectation propagation for approximate Bayesian inference. *Uncertainty in AI'01*. `http://www.stat.cmu.edu/~minka/papers/ep/`.

Minka, T. P. (2004). Power EP. `http://www.stat.cmu.edu/~minka/`.

Minka, T. P., & Lafferty, J. (2002). Expectation propagation for the generative aspect model. *Proc UAI*.

Murray, I., & Ghahramani, Z. (2004). Bayesian learning in undirected graphical models: Approximate MCMC algorithms. *UAI*.

Qi, Y. (2004). *Extending expectation propagation for graphical models*. Doctoral dissertation, MIT.

Qi, Y., Minka, T. P., Picard, R. W., & Ghahramani, Z. (2004). Predictive automatic relevance determination by expectation propagation. *Proceedings of Twenty-first International Conference on Machine Learning*.

Sha, F., & Pereira, F. (2003). *Parsing with conditional random fields* (Technical Report MS-CIS-02-35). University of Pennsylvania.

Taskar, B., Guestrin, C., & Koller, D. (2004). Max-margin Markov networks. In S. Thrun, L. Saul and B. Schölkopf (Eds.), *Advances in neural information processing systems 16*.

Wiegerinck, W., & Heskes, T. (2002). Fractional belief propagation. *NIPS 15*.

Poisson-Networks: A Model for Structured Point Processes

Shyamsundar Rajaram
ECE Department
University of Illinois
Urbana, USA

Thore Graepel
MLP Group
Microsoft Research
Cambridge, UK

Ralf Herbrich
MLP Group
Microsoft Research
Cambridge, UK

Abstract

Modelling structured multivariate point process data has wide ranging applications like understanding neural activity, developing faster file access systems and learning dependencies among servers in large networks. In this paper, we develop the Poisson network model for representing multivariate structured Poisson processes. In our model each node of the network represents a Poisson process. The novelty of our work is that waiting times of a process are modelled by an exponential distribution with a piecewise constant rate function that depends on the event counts of its parents in the network in a generalised linear way. Our choice of model allows to perform exact sampling from arbitrary structures. We adopt a Bayesian approach for learning the network structure. Further, we discuss fixed point and sampling based approximations for performing inference of rate functions in Poisson networks.

1 Introduction

Structured multivariate point processes appear in many different settings ranging from multiple spike train recordings, file access patterns and failure events in server farms to queuing networks. Inference of the structure underlying such multivariate point processes and answering queries based on the learned structure is an important problem. For example, learning the structure of cooperative activity between multiple neurons is an important task in identifying patterns of information transmission and storage in cortical circuits [Brillinger and Villa, 1994, Aertsen et al., 1989, Oram et al., 1999, Harris et al., 2003, Barbieri et al., 2001, Brown et al., 2004]. Similarly, learning the access patterns of files can be exploited for building faster file access systems.

Consider V time series of events $\mathbf{t}_i$. We would like to find a compact representation of the joint probability distribution of the V time series. Assuming each time series is modelled by an inhomogeneous Poisson process, a unique representation is obtained in terms of V rate functions $\lambda_i(t)$ each of which depends on all events of the V times series up to time t. Clearly, we need further assumption on the rate functions to infer the rate functions from a finite amount of data. The proposed approach makes four crucial assumptions:

1. The rate function of each process depends only on the history in a short time window into the past.
2. The rate function of each process depends only on a small number of other processes. Adopting a directed graph notation, these are also referred to as *parent* nodes.
3. The rate function of each process depends only on the empirical rates of its parents.
4. The rate function of each process is parameterised by a generalised linear model.

The standard approach adopted in modelling such time series is by discretising the time axis into intervals of fixed length δ and transforming the time series into a sequence of counts per interval. The dependency structure between nodes of such a network, which is also known as a *dynamic Bayesian network* (DBN) [Dean and Kanazawa, 1989, Murphy, 2001], can be modelled using transition probabilities between states at time t and $t + \delta$ given the states of all the parents of each node at time t. This approach suffers from the problem that the discretisation is somewhat arbitrary: A small value of δ will result in a redundant representation and a huge computational overhead but a large value of δ may smooth away important details in the data.

We overcome the issues of discretisation by considering the limit $\delta \to 0$ and thus modelling the waiting time in each time series directly. Nodelman et al. [2002] and Nodelman et al. [2003] use a similar continuous time approach for modelling homogeneous Markov processes with a finite number of states. In their work the rate functions depend on the current state of the parents only which leads to an efficient and exact learning algorithm. In our setup, we model the waiting times as an exponential distribution with piecewise constant rates that depend on the count history of parent nodes.

The paper is structured as follows: In Sec. 2 we introduce our Poisson network model. In Sec. 3 we describe an efficient technique for performing exact sampling from a Poisson network. We describe parameter estimation and structure learning using approximate Bayesian inference techniques in Sec. 4. In Sec. 5 we discuss approximate marginalisation and inference based on sampling and fixed point methods. Finally, we describe experiments on data generated by our sampling technique for performing parameter learning, structure estimation and inference in Sec. 6.

2 The Poisson Network Model

Consider time series data $\mathbf{t}_i \in (\mathbb{R}^+)^{N_i}$ from V point processes. Each element of $\mathbf{t}_i = [t_{i,1}, \ldots, t_{i,N_i}]$ corresponds to series of times $t_{i,j}$ at which a particular event occurred. We model each time series as an inhomogeneous Poisson process and the modelling problem is to capture the dependency between different processes, both qualitatively (structure) and quantitatively (parameters).

A Poisson process is an instance of a counting process which is characterised by a rate function $\lambda(t)$. If the rate function is constant over time, the Poisson process is called *homogeneous*, otherwise it is called *inhomogeneous* [Papoulis, 1991]. A very useful property of a homogeneous Poisson process is that the waiting time between two consecutive events is exponentially distributed with rate λ, that is,

$$\forall t \in \mathbb{R}^+ : p(t|\lambda) := \lambda \exp(-\lambda t)\,.$$

Note that the mean of the waiting time distribution is given by λ^{-1}.

The waiting time distribution for a non-homogeneous process is a generalized exponential distribution. In the special case of a piecewise constant rate function $\lambda(t)$, the waiting time has a piecewise exponential distribution.

Proposition 1 (Piecewise Exponential Distribution). *Suppose we are given l rates $\boldsymbol{\lambda} \in (\mathbb{R}^+)^l$ and l sets $\mathcal{T}_k$ of non-overlapping intervals where the average waiting time in each of the intervals T in $\mathcal{T}_k$ is governed by λ_k. The waiting time of the piecewise exponential distribution has the following density:*

$$\begin{aligned} p(t|\lambda, \mathcal{T}) &:= \prod_{k=1}^{l} {\lambda_k}^{a_k(t,\mathcal{T}_k)} \exp(-\lambda_k b_k(t, \mathcal{T}_k)), \\ a_k(t, \mathcal{T}_k) &:= \sum_{[t_0,t_1)\in\mathcal{T}_k} \mathbb{I}_{t\in[t_0,t_1)}, \\ b_k(t, \mathcal{T}_k) &:= \sum_{[t_0,t_1)\in\mathcal{T}_k} (t_1 - t_0)\mathbb{I}_{t>t_1} + (d - t_0)\mathbb{I}_{t\in[t_0,t_1)}. \end{aligned}$$

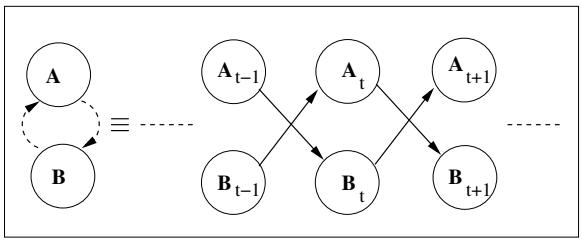


Figure 1: Illustration of a Poisson network with a cycle unwrapped in time. Note that we use dashed lines to indicate parent relationships in Poisson networks. These arrows should always be interpreted as pointing forward in time.

In the Poisson network model we aim at modelling the independence relations between the V processes. Similarly to Bayesian networks, the independence relations are implicitly represented by the connectivity of a directed graph. Hence, the network structure can be fully represented by all parent relationships, $M := \{\pi(i) \subseteq \{1, \ldots, V\}\}$. Interestingly, the semantics of a parent relationship is slightly different from Bayesian networks: Since the rate function of each node only depends on the past history of its parents, cycles w.r.t. the parent set M are permissible (see Figure 1).

We model the dependency of a node i on its parents $\pi(i) = \{p_1, \ldots, p_{m_i}\}$ by constraining the rate function $\lambda_i(t)$ to be a function of the event counts of all parents in time windows of length ϕ, $\mathbf{n}_i(t) := [n_{i,p_1}(t), \ldots, n_{i,p_{m_i}}(t)]$ where n_{i,p_j} represents the number of events of node p_j in the time window $[t-\phi, t)$.[1]

We consider a generalized linear model for the rate function $\lambda(t)$ in terms of the counts $\mathbf{n}_i(t)$. Possible link functions include the probit or sigmoid function which exhibit a natural saturation characteristic. However, in the following we will consider the canonical link

[1] Note that we will treat ϕ as a fixed quantity throughout the paper.

function for the Poisson model resulting in:

$$\lambda_i(t; \mathbf{w}_i, \mathbf{x}_i) = \exp\left(w_{i,0} + \sum_{j \in \pi(i)} w_{i,j} x_{i,j}(t)\right), \quad (1)$$

where $w_{i,0}$ represents a bias term and translates into a multiplicative base rate $\exp(w_{i,0})$. We define

$$\begin{aligned} x_{i,j}(t) &:= \ln\left(1 + \hat{\lambda}_{i,j}(t)\right), \\ \hat{\lambda}_{i,j}(t) &:= \frac{n_{i,j}(t)}{\phi}. \end{aligned}$$

Note that $\hat{\lambda}_{i,j}(t)$ is the *empirical* rate of node j w.r.t. node i. Alternatively, the rate function can be written in a way that is more amenable for inference (see Sec. 5):

$$\lambda_i(t) = \exp(w_{i,0}) \prod_{j \in \pi(i)} \left(1 + \hat{\lambda}_{i,j}(t)\right)^{w_{i,j}}.$$

A positive value for $w_{i,j}$ indicates an excitatory effect and a negative value corresponds to an inhibitory effect on the rate. The rate of each node at any given time instant is a function of the empirical rate of its parents resulting in an inhomogeneous process. Note that in practice there are only finitely many different count vectors due to the finite length time windows. The piecewise constant characteristic of the rate function and the absence of cycles in the Poisson network enables us to do exact sampling (see Sec. 3).

Let us derive the probability distribution of the time series corresponding at given nodes. Consider the $l-1^{\text{th}}$ and l^{th} events occurring at times t_{l-1} and t_l in a given node and the instants at which the count vector changes in this interval be represented as $\breve{t}_{l,1}, \ldots, \breve{t}_{l,k_l}$.[2] The probability density of an event at time t_l given the previous event happened at t_{l-1} is denoted by $p(t_l|t_{l-1}, \mathbf{w}, M)$. This density is a product of probabilities of non-occurrence of an event in each of the disjoint subintervals, i.e. $(t_{l-1}, \breve{t}_{l,1}], (\breve{t}_{l,1}, \breve{t}_{l,2}], \ldots, (\breve{t}_{l,k_l-1}, \breve{t}_{l,k_l}]$ and the probability density of occurrence of an event at t_l in the subinterval $(\breve{t}_{l,k_l}, t_l]$:

$$p(t_l|t_{l-1}, \mathbf{w}, M) = \lambda_{l,k_l+1} \left(\prod_{j=1}^{k_l+1} \exp\left(-\lambda_{l,j}\tau_{l,j}\right)\right),$$

where $\lambda_{l,j}$ is the rate as in (1) for a node which is a function of the event counts of its parents in the jth subinterval corresponding to the lth event and the durations $\tau_{l,j}$ are defined by

$$\begin{aligned} \tau_{l,1} &:= \breve{t}_{l,1} - t_{l-1}, \\ \tau_{l,j} &:= \breve{t}_{l,j} - \breve{t}_{l,j-1}, \quad j \in \{2, \ldots, k_l\} \\ \tau_{l,k_l+1} &:= t_l - \breve{t}_{l,k_l}. \end{aligned}$$

The probability density for the time series $\mathbf{t} = [t_1, \ldots, t_N]$ of a given node is obtained as the product of the probability densities for each of the mutually exclusive subintervals $(t_0, t_1], \ldots, (t_{N-1}, t_N]$,

$$p(\mathbf{t}|\mathbf{w}, M) = \prod_{l=1}^{N} p(t_l|t_{l-1}, \mathbf{w}, M), \quad (2)$$

where the initial time t_0 is assumed to be 0.

[2] We drop the node subscript and suppress the dependence on the time series of all parent nodes for better readability.

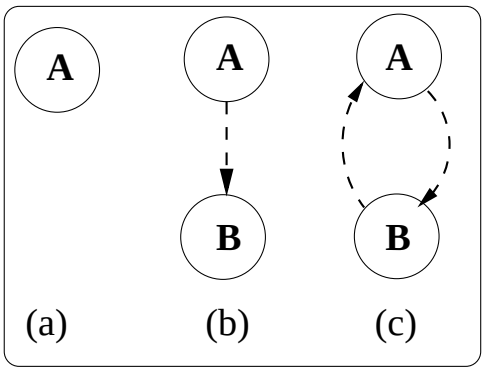


Figure 2: Poisson networks for illustrating sampling.

3 Sampling from a Poisson Network

The piecewise constant behavior of the rate function $\lambda(t)$ and the absence of cycles in the network allows us to perform exact sampling from a Poisson Network. Let us consider the simple case of sampling from a single node network as shown in Fig. 2 (a). Sampling is straightforward in this case, because the node represents a homogeneous Poisson process. The standard way to sample from a homogeneous Poisson process is to make use of the property that the waiting times between two adjacent events are iid and are distributed exponentially with rate parameter λ.

Let us consider a two node network shown in Fig. 2 (b). Sampling for node A can be done in a straightforward way as explained above because at any time t, A_t is independent of B_t. The rate function for node B can be calculated using (1) once the whole time series for node A is known. The rate function for B is piecewise constant because of the finite number of events for node A in the time window $[t-\phi, t]$ for varying t. Sampling from a waiting time distribution with a

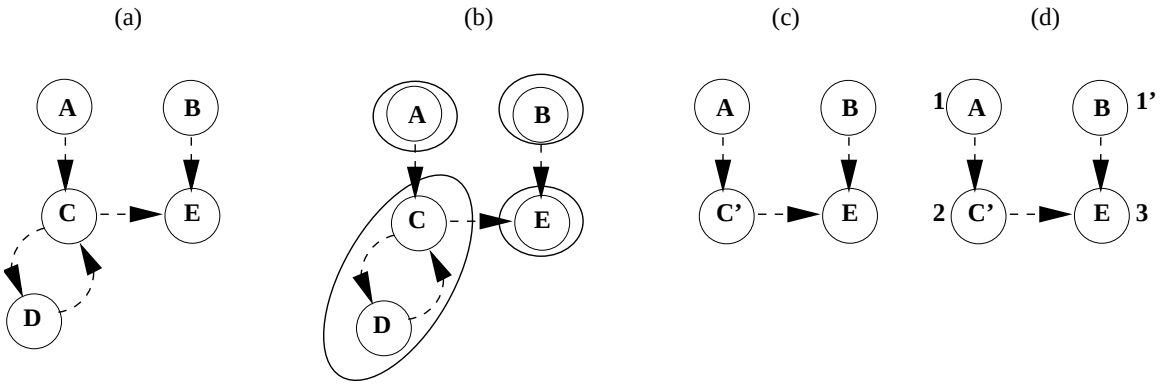


Figure 3: (a) An arbitrary Poisson network. (b) SCC's indicated by ellipses. (c) Directed acyclic graph C' represents the SCC formed by nodes C and D. (d) Topological ordering where the number ($1'$ and 1 can be sampled independently of each other in parallel) written adjacent to the node indicates the order.

piecewise constant rate function is done by rejection sampling. Denote the current value of the rate function by $\lambda(t)$ and assume the rate remains unchanged until time $\hat{t}$. Sample τ from the waiting time distribution with parameter $\lambda(t)$. Accept τ if $t + \tau \leq \hat{t}$ and reject τ otherwise. The sampling time is now updated to $\hat{t}$ or $t + \tau$ depending on whether the sample is rejected or accepted, respectively.

Now, let us consider a two node network as shown in Fig. 2 (c) where there exists a cyclic relationship between nodes A and B. It is worth mentioning again that node B depends on the event counts of node A in the past only and vice versa. Sampling from the nodes cannot be done in an independent way because of the cyclic relationship. Let the values $\lambda_A(t)$ and $\lambda_B(t)$ of the rate functions for nodes A and B, respectively, be calculated by (1) and assume that the rates remain constant until $\hat{t}_A$, $\hat{t}_B$ (excluding the mutual interaction in the future). Sample waiting times τ_A and τ_B for both nodes using the rates $\lambda_A(t)$ and $\lambda_B(t)$, respectively. The sample corresponding to $\max(\tau_A, \tau_B)$ is rejected because an earlier event at the other node might have changed the value of the rate function; the other sample is accepted if it is within the constant time interval of the firing rate. The sampling time is updated to $\min(\hat{t}_A, \hat{t}_B)$ or $t + \min(\tau_A, \tau_B)$ depending on whether the other sample was rejected or accepted, respectively.

This sampling technique can be generalised to an arbitrary number of nodes. We note that the structure of the network can be made use of in order to perform sampling in a very efficient way. The two key observations are that,

1. Certain groups of nodes have to sampled from in a synchronized, dependent way because of mutual dependence of nodes in the group.
2. Sampling has to be done in a certain order.

The first observation is illustrated in Fig. 3 (b) and such groups of nodes are known as strongly connected components (SCC) in graph theory [Cormen et al., 2001]. A strongly connected component is a directed subgraph with the property that for all ordered pairs of vertices (u, v), there exists a path from u to v. A variant of depth first search can be used to find all the SCCs of a graph. A directed acyclic graph graph is obtained by replacing each SCC by a single node as shown in Fig. 3 (c). Now we observe that the nodes of the directed acyclic graph can be sampled from in such a way that, when sampling a given node, its parents have been sampled before (see Fig. 3 (d)). Topological sorting is a method to compute such an order efficiently (see Cormen et al. [2001]).

4 Parameter Estimation and Structure Learning

Given time series data $\mathbf{T} := [\mathbf{t}_1, \ldots, \mathbf{t}_V]$ we are interested in finding the most plausible structure M^* and parameter estimates for $\mathbf{w}$ in the linear model of the rate functions (see (1)). We take a Bayesian approach to the problem of parameter estimation which, at the same time, provides a score over structure M by the marginalised likelihood. We choose a Gaussian prior distribution over the weights

$$p(\mathbf{W}|M) = \prod_{i=1}^{V} \mathcal{N}(\mathbf{w}_i; \mathbf{0}, \sigma^2 \mathbf{I}) \,.$$

Using Bayes rule, we obtain the posterior

$$p(\mathbf{W}|\mathbf{T}, M) = \frac{p(\mathbf{T}|\mathbf{W}, M) p(\mathbf{W}|M)}{p(\mathbf{T}|M)} \,.$$

The structure learning problem can be written as,

$$\begin{aligned} M^* &:= \operatorname*{argmax}_M \; p(M|\mathbf{T}) \\ &= \operatorname*{argmax}_M \; p(\mathbf{T}|M)p(M) \\ p(\mathbf{T}|M) &= \prod_{i=1}^{V} p(\mathbf{t}_i|\{\mathbf{t}_j|j \in \pi(i)\}) \\ p(M) &:= \prod_{i=1}^{V} p(\pi(i)) \,, \end{aligned}$$

where we make use of a structural prior $p(M)$ that factors over the parents of the V nodes. Note that $p(\mathbf{t}_i|\{\mathbf{t}_j|j \in \pi(i)\})$ corresponds to (2) marginalised over $\mathbf{w}$. In contrast to structure learning in Bayesian networks which requires optimisation over the set of directed acyclic graphs, no such constraints are imposed in Poisson network or continuous time Bayesian networks [Nodelman et al., 2003]. Hence, the structure learning problem can be solved by finding the most likely parents of each node independently. In theory the exact structure can be learned without regard to the number of parents. However, in practice the running time of the learning procedure is a function of the number of parents of a node and hence while learning structures we fix the maximum number of parents. Now we present two approximate Bayesian inference techniques for parameter estimation and structure learning.

4.1 Laplace Approximation

In the Laplace approximation [Kass and Raftery, 1995], the posterior is approximated by a multivariate Gaussian density,

$$p(\mathbf{w}_i|\mathbf{T}, M) \approx \mathcal{N}(\mathbf{w}_i; \mathbf{w}_{\text{MAP}}, \boldsymbol{\Sigma}) \,,$$

where $\mathbf{w}_{\text{MAP}}$ is the mode of the posterior,

$$\mathbf{w}_{\text{MAP}} := \operatorname*{argmax}_{\mathbf{w}_i} p(\mathbf{T}, \mathbf{w}_i|M) \,,$$

and the covariance matrix $\boldsymbol{\Sigma}$ is given by

$$\boldsymbol{\Sigma} := \left(-\nabla\nabla^T_{\mathbf{w}_i} \; \ln(p(\mathbf{T}, \mathbf{w}_i|M))|_{\mathbf{w}=\mathbf{w}_{\text{MAP}}}\right)^{-1} .$$

The marginalised likelihood can be obtained by,

$$\begin{aligned} p(\mathbf{T}|M) &= \int p(\mathbf{T}, \mathbf{w}_i|M) \; d\mathbf{w}_i \\ &\approx \int \sqrt{(2\pi)^{m_i}|\boldsymbol{\Sigma}|}\mathcal{N}(\mathbf{w}_i; \mathbf{w}_{\text{MAP}}, \boldsymbol{\Sigma}) \; d\mathbf{w}_i \\ &= \sqrt{(2\pi)^{m_i}|\boldsymbol{\Sigma}|} \,. \end{aligned}$$

The mode $\mathbf{w}_{\text{MAP}}$ is found by the conjugate gradients algorithm using the gradient of $\ln(p(\mathbf{T}, \mathbf{w}_i|M))$ w.r.t. $\mathbf{w}$ (see also (2)).

4.2 Variational Approach

The variational approach to solving problem of parameter estimation and structure learning is to introduce a family of probability distribution $q_{\boldsymbol{\theta}}(\mathbf{w}_i)$ parameterised over $\boldsymbol{\theta}$ which gives rise to a lower bound on the marginalised likelihood[3]:

$$\ln(p(\mathbf{T}|M)) \geq \mathrm{L}_M(\boldsymbol{\theta}) \,,$$

$$\mathrm{L}_M(\boldsymbol{\theta}) := \left\langle \ln\left(\frac{p(\mathbf{w}_i|M)}{q_{\boldsymbol{\theta}}(\mathbf{w}_i)}\right)\right\rangle_{q_{\boldsymbol{\theta}}} + \langle \ln(p(\mathbf{T}|\mathbf{w}_i, M))\rangle_{q_{\boldsymbol{\theta}}} \,,$$

which follows from an application of Jensen's inequality to the concave logarithm function [Jordan et al., 1999]. This lower bound on the log-marginal probability is a function of the parameters $\boldsymbol{\theta}$ of the distribution q. This bound is tight if we choose $q_{\boldsymbol{\theta}}(\mathbf{w}_i)$ to be the true posterior $p(\mathbf{w}_i|\mathbf{T}, M)$. We also observe that the lower bound is the sum of two terms which correspond to the negative of the Kullback-Leibler divergence between the prior term and the distribution $q_{\boldsymbol{\theta}}(\mathbf{w}_i)$ and the second term is the log-likelihood of the data averaged over $q_{\boldsymbol{\theta}}(\mathbf{w}_i)$.

The idea of the variational approximation Bayesian algorithm is to maximize the lower bound L_M with respect to the parameters of the q distribution. The structure learning problem can be posed as maximization of the lower bound $\mathrm{L}_M(\boldsymbol{\theta})$ which can be written as,

$$M^* := \operatorname*{argmax}_M \left[\max_{\boldsymbol{\theta}} \mathrm{L}_M(\boldsymbol{\theta})\right] .$$

Similar to the Laplace approximation, we choose $q_{\boldsymbol{\theta}}(\mathbf{w}_i)$ to be a multivariate Gaussian $\mathcal{N}(\mathbf{w}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma})$. For a given network structure M, we can use conjugate gradients to find the maximum of $\mathrm{L}_M(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Note that the gradients are given by

$$\begin{aligned} \nabla_{\boldsymbol{\mu}}(\mathrm{L}_M(\boldsymbol{\mu}, \boldsymbol{\Sigma})) &= -\boldsymbol{\mu} + \sum_{i=1}^{N} \mathbf{x}_{i,k_i} \ldots \\ &\quad - \sum_{i=1}^{N}\sum_{j=1}^{k_i} \tau_{i,j}\mathbf{x}_{i,j} \exp\left(\frac{\mathbf{x}^T_{i,j}\boldsymbol{\Sigma}\mathbf{x}_{i,j} + 2\boldsymbol{\mu}^T\mathbf{x}_{i,j}}{2}\right) , \\ \nabla_{\boldsymbol{\Sigma}}(\mathrm{L}_M(\boldsymbol{\mu}, \boldsymbol{\Sigma})) &= -\frac{1}{2\sigma^2}\mathbf{I} + \frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{\Sigma})^{-\mathbf{T}} \ldots \\ &\quad - \sum_{i=1}^{N}\sum_{j=1}^{k_i} \frac{1}{2}\tau_{i,j}\mathbf{x}_{i,j}\mathbf{x}^T_{i,j} \exp\left(\frac{\mathbf{x}^T_{i,j}\boldsymbol{\Sigma}\mathbf{x}_{i,j} + 2\boldsymbol{\mu}^T\mathbf{x}_{i,j}}{2}\right) . \end{aligned}$$

5 Inference in a Poisson network

Once the network structure and parameters are learned, several queries regarding the distribution represented by the network can be answered. A common

[3] We use the shorthand notation $\langle f(\cdot)\rangle_q$ to denote an expectation of f w.r.t. the distribution q.

inference problem that is encountered in a Bayesian network is one where data is available for some nodes (denoted by M_D), there are a few query nodes (denoted by M_Q) whose behavior has to be estimated from the data and the remaining nodes are hidden nodes (denoted by M_H for which no data is available. We pose an analogous problem for Poisson networks. The rate of any arbitrary node can be computed if the time series data for all its parents are known for the period of interest. However, certain configurations of the problem involving hidden nodes are not easy to solve because of the problem of *entanglement* as mentioned in Nodelman et al. [2002]. The standard procedure in a Bayesian network is to marginalise over all the hidden nodes and obtain an estimation for the query nodes using the data in observed nodes. Marginalising over a hidden node amounts to integrating over all possible time series for a particular node which, in general, is impossible. Hence, approximations are necessary.

5.1 Inference by Sampling

A straightforward way to solve the inference problem is to perform marignalisation of the hidden nodes by using a few instantiations of them which can be obtained by sampling from the network. The samples can then be used to obtain averaged rate estimates at each of the query nodes. The sampling procedure is the same as our procedure in Sec. 3 with a minor modification that the sampling is to be performed conditioned on the data that has been observed in the data nodes M_D. We observe that if the data for a node i is completely known, then the parent nodes of i do not have any influence on i in the subsequent sampling process. Hence, we can safely remove all the parental relationships for all the fully observed nodes and obtain a new network $M' = M - \{i \to j, i \in M_D, j \in \pi(i)\}$. Sampling is done from the new network M' for all the unobserved nodes and then average firing rates can be obtained for all the query nodes.

5.2 Estimation of Steady State Rate

An alternative way to solve the inference problem is to approximate the empirical rate $\hat{\lambda}$ with a steady state rate. According to our model, the rate of a node can be written as,

$$\lambda_i(t) = \exp\left(w_{i,0} + \sum_{j=1}^{V}\left(w_{i,j}\ln\left(1+\hat{\lambda}_{i,j}(t)\right)\right)\right) . \quad (3)$$

We notice that this is (1) if we consider $w_{i,j} = 0$ for all the pairs of nodes which are not dependent. The empirical rate $\hat{\lambda}_{i,j}(t)$ cannot be obtained unless the whole time series is observed. Hence, we approximate the empirical rate by the true rate,

$$\hat{\lambda}_{i,j}(t) = \frac{n_{i,j}(t)}{\phi} \approx \frac{1}{\phi}\int_{t-\phi}^{t}\lambda_j(\tilde{t})d\tilde{t} \approx \lambda_j(t)\,,$$

where we assume that the length of time window, ϕ, is very small. Substituting back in (3) we obtain

$$\ln(\lambda_i(t)) \approx w_{i,0} + \sum_{j=1}^{V} w_{i,j}\ln\left(1+\lambda_j(t)\right). \quad (4)$$

We make the assumption that the rate is constant in the time interval $[t-\phi, t]$ and hence the Poisson process of each of the parents $j, j \in \{1,\ldots,V\}$ is assumed to be a homogeneous Poisson process with rate $\lambda_j(t)$. Now, (4) can be constructed for all nodes that are not observed and the set of equations have the form $\boldsymbol{\lambda}(t) = F(\boldsymbol{\lambda}(t))$, $F : \mathbb{R}^V \to \mathbb{R}^V$, whose solution are the fixed points of the system. We perform fixed point iterations starting from randomly initialized values for $\lambda_i(t), i \in \{1 \ldots V\}$. In experiments we observe that at convergence, the estimated rate corresponds to the mean rate in the time interval $[t-\phi, t]$ calculated from actual data.

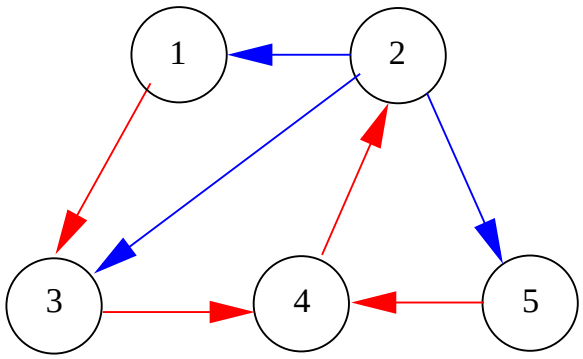


Figure 4: The random Poisson network graph that generated the samples in Fig. 5. Red arrow indicates an inhibitory influence and the blue arrow indicates an excitatory influence.

6 Experimental Results

In this section, we test the presented methods on data sampled using the algorithms from Sec. 3. We show experiments of approximate inference of rate using a sampling based approximation and a fixed point approximation.

Sampling Firstly, we generate a random graph (see Fig. 4) with V nodes. As mentioned before, because of computational issues we fix the maximum number of

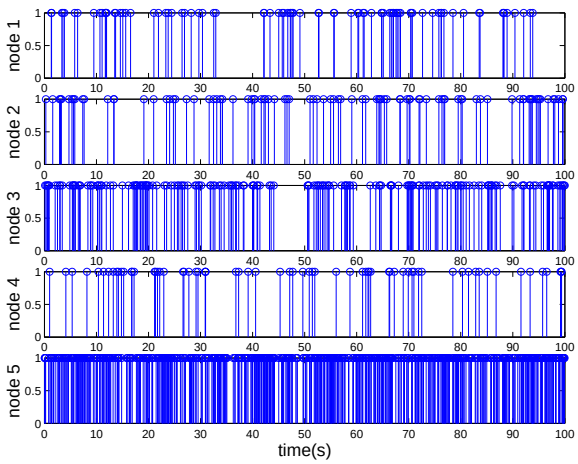


Figure 5: Samples generated from a random network with $V = 5$ and maximum number of parents is 2

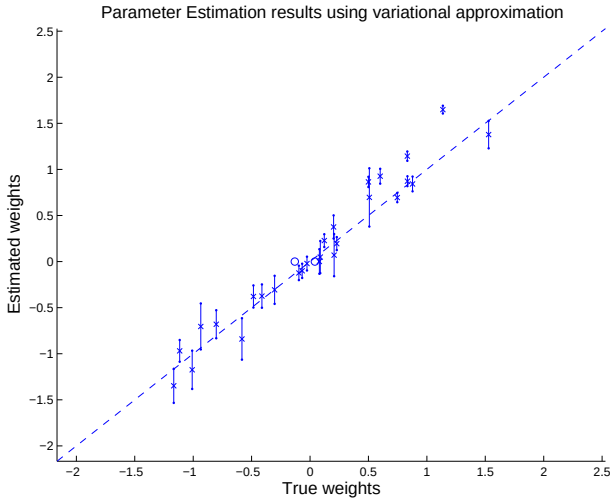


Figure 6: Results for parameter estimation and structure learning using the variational approximation for a network with 15 nodes in which maximum number of parents is 2. Each node i with a parent j has a true expected weight $w_{i,j}$ (x-axis) and a posterior estimate (y-axis) shown as a 5%-95% posterior quantile interval with a mark indicating the mean. The empty circles indicate the edges that were not identified by the structure learning algorithm.

parents $\pi_{\max}$ for every node. Now, our efficient sampling technique given in Sec. 3 is used to generate samples from the network. Samples generated from a randomly generated network with $V = 5$ and $\pi_{\max} = 2$ is as shown in Fig. 5. The time window duration ϕ was fixed to 1 second.

Parameter estimation and structure learning The Laplace and variational approximation developed in Sec. 4.1 and Sec. 4.2 are tested using samples generated from random graph structures. We observed that the posterior distribution $p(\mathbf{w}_i|\mathbf{T}, M)$ can be approximated very well by a multivariate Gaussian. Thus, both approximations methods perform very accurately. Fig. 6 shows the parameter estimation and structure learning results obtained using variational approximations for a 15 node network with maximum number of parents restricted to 2. The results indicate that the few edges that were missed by the structure learning algorithm (circles) correspond to weak dependencies i.e., edges with weights close to zero. The results of the Laplace approximation is similar to the variational approximation except that the variational approximation had higher confidence in its estimate.

Approximate Inference of rates The approximate inference techniques developed in Sec. 5 was tested on a random graph having 10 nodes. Samples were generated from the network using our sampling technique. Nodes were chosen at random and marked as observed, hidden and as query nodes. The inference task was to estimate rates for all the nodes marked as query nodes. The samples generated for the observed nodes were made use of to perform inference of rates using the sampling based approximation and the fixed point approximation. The results shown in Fig. 7 shows that the fixed point approximation technique (which is significantly faster than the sampling based approximation) closely tracks the true rate.

7 Conclusions and Discussion

Poisson networks are models of structured multivariate point processes and have the potential to be applied in many fields. They are designed such that sampling and approximate learning and inference are tractable using the approaches described above. In particular, structure learning is carried out in a principled yet efficient Bayesian framework.

Future applications of the Poisson network model include biological problems such as analysing multiple neural spike train data Brown et al. [2004] as well as application in computer science such as prediction of file access patterns, network failure analysis and queuing networks.

A promising direction for future research is to combine Poisson networks with continuous time Bayesian networks in order to be able to model both event counts and state (transitions). For example, consider file access events and the running states of CPU processes. Clearly, they exhibit an interesting relationship the discovery of which would it possible to predict and

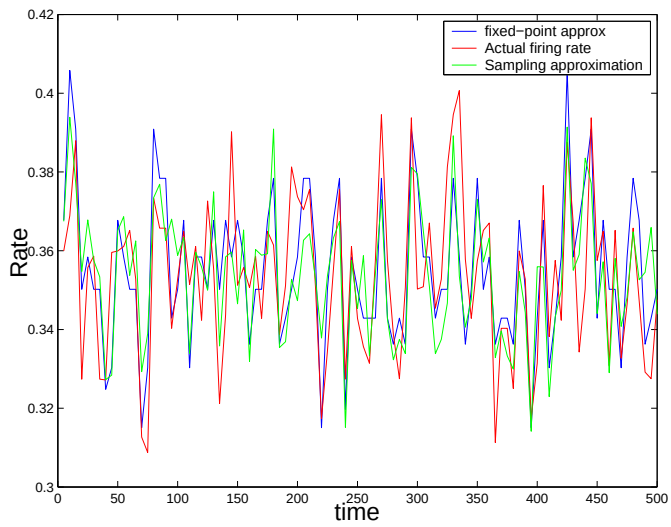


Figure 7: Inference of rate functions

hence optimise process-file interactions. Similarly, in biological application external stimuli can be modelled as states influencing physiological events such as neural spike activity.

Finally, it would be of great interest to study the information processing potential of Poisson networks in the sense of artificial neural networks (see Barber [2002]).

Acknowledgments Shyamsundar would like to thank Microsoft Research for providing funding for this project during an internship in Cambridge and Thoams Huang for generous support. We are very indebted to Ken Harris for interesting discussions about biological applications of Poisson networks, and we would like to thank Tom Minka for interesting discussions about approximate inference.

References

A. M. Aertsen, G. L. Gerstein, M. K. Habib, and G. Palm. Dynamics of neuronal firing correlation: modulation of "effective connectivity". *Journal of Neurophysiology*, 61(5):900–917, 1989.

D. Barber. Learning in spiking neural assemblies. In *Advances in Neural Information Processing Systems*, pages 165–172, 2002.

R. Barbieri, M. C. Quirk, L. M. Frankb, M. A. Wilson, and E. N. Browna. Construction and analysis of non-poisson stimulus-response models of neural spiking activity. *Journal of Neuroscience Methods*, 105(1):25–37, 2001.

D. R. Brillinger and A. E. P. Villa. Examples of the investigation of neural information processing by point process analysis. *Advanced Methods of Physiological System Modelling*, 3:111–127, 1994.

E. N. Brown, R. E. Kass, and P. P. Mitra. Multiple neural spike train data analysis: state-of-the-art and future challenges. *Nature Neuroscience*, 7:456–461, April 2004.

T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. MIT Press, 2001.

T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150, 1989.

K. D. Harris, J. Csicsvari, H. Hirase, G. Dragoi, and G. Buzsaki. Organization of cell assemblies in the hippocampus. *Nature*, 424:552–556, July 2003.

M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

R. E. Kass and A. E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90:773–795, 1995.

K. P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, Computer Science Division, 2001.

U. Nodelman, C. Shelton, and D. Koller. Continuous time Bayesian networks. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 378–387, San Francisco, CA, 2002. Morgan Kaufmann Publishers.

U. Nodelman, C. Shelton, and D. Koller. Learning continuous time Bayesian networks. In *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*, pages 451–458, San Francisco, CA, 2003. Morgan Kaufmann Publishers.

M. W. Oram, M. C. Wiener, R. Lestienne, and B. J. Richmond. Stochastic nature of precisely timed spike patterns in visual system neuronal responses. *Journal of Neurophysiology*, 81(6):3021–3033, 1999.

A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. Mcgraw-Hill, New York, 3 edition, 1991.

Deformable Spectrograms

Manuel Reyes-Gomez
LabROSA
Department of Electrical Engineering
Columbia University
mjr59@ee.columbia.edu

Nebojsa Jojic
Microsoft Research
One Microsoft Way
Redmond,WA.
jojic@microsoft.com

Daniel P.W. Ellis
LabROSA
Department of Electrical Engineering
Columbia University
dpwe@ee.columbia.edu

Abstract

Speech and other natural sounds show high temporal correlation and smooth spectral evolution punctuated by a few, irregular and abrupt changes. In a conventional Hidden Markov Model (HMM), such structure is represented weakly and indirectly through transitions between explicit states representing 'steps' along such smooth changes. It would be more efficient and informative to model successive spectra as *transformations* of their immediate predecessors, and we present a model which focuses on local deformations of adjacent bins in a time-frequency surface to explain an observed sound, using explicit representation only for those bins that cannot be predicted from their context. We further decompose the log-spectrum into two additive layers, which are able to separately explain and model the evolution of the harmonic excitation, and formant filtering of speech and similar sounds. Smooth deformations are modeled with hidden transformation variables in both layers, using Markov Random fields (MRFs) with overlapping subwindows as observations; inference is efficiently performed via loopy belief propagation. The model can fill-in deleted time-frequency cells without any signal model, and an entire signal can be compactly represented with a few specific states along with the deformation maps for both layers. We discuss several possible applications for this new model, including source separation.

1 Introduction

Hidden Markov Models (HMMs) work best when only a limited set of distinct states need to be modeled, as in the case of speech recognition where the models need only be able to discriminate between phone classes. When HMMs are used with the express purpose of accurately modeling the full detail of a rich signal such as speech, they require a large number of states. In [1](Roweis 2000), HMMs with 8,000 states were required to accurately represent one person's speech for a source separation task. The large state space is required because it attempts to capture every possible instance of the signal. If the state space is not large enough, the HMM will not be a good generative model since it will end up with a "blurry" set of states which represent an average of the features of different segments of the signal, and cannot be used in turn to "generate" the signal.

In many audio signals including speech and musical instruments, there is a high correlation between adjacent frames of their spectral representation. Our approach consists of exploiting this correlation so that explicit models are required only for those frames that cannot be accurately predicted from their context. In [2](Bilmes 1998), context is used to increase the modelling power of HMMs, while keeping a reasonable size parameters space, however the correlation between adjacent frames is not explicity modeled. Our model captures the general properties of such audio sources by modeling the evolution of their harmonic components. Using the common source-filter model for such signals, we devise a layered generative graphical model that describes these two components in separate layers: one for the excitation harmonics, and another for resonances such as vocal tract formants. This layered approach draws on successful applications in computer vision that use layers to account for different sources of variability [3, 4, 5](Jojic 2001,Levin 2002,Jojic 2003). Our approach explicitly models the self-similarity and dynamics of each layer by fitting the log-spectral representation of the signal in frame t with a set of transformations of the log-spectra in frame $t-1$. As a result, we do not require separate states for every possible spectral configuration, but only a limited set of "sharp" states that can still cover the full spectral variety of a source via such transformations. This approach is thus suitable for any time series data with high correlation between adjacent observations.

We will first introduce a model that captures the spectral de-

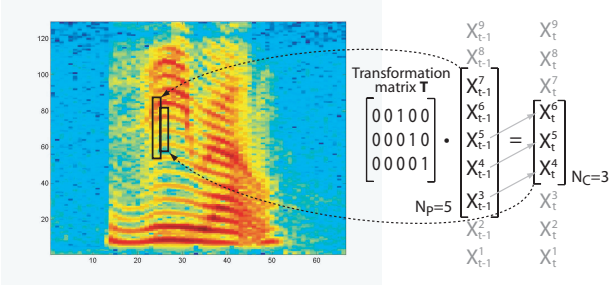


Figure 1: The $N_C = 3$ patch of time-frequency bins outlined in the spectrogram can be seen as an "upward" version of the marked $N_P = 5$ patch in the previous frame. This relationship can be described using the matrix shown.

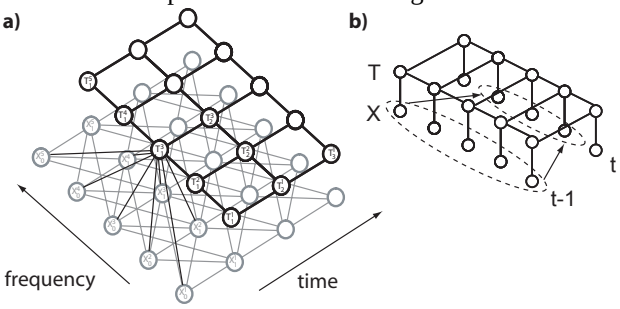


Figure 2: a) Graphical model b) Graphical simplification.

formation field of the speech harmonics, and show how this can be exploited to interpolate missing observations. Then, we introduce the two-layer model that separately models the deformation fields for harmonic and formant resonance components, and show that such a separation is necessary to accurately describe speech signals through examples of the missing data scenario with one and two layers. Then we will present the complete model including the two deformation fields and the "sharp" states. This model, with only a few states and both deformation fields, can accurately reconstruct the signal. This paper fully describes the operation and implementation of this complete model, which was only described as future work in [6](Reyes-Gomez 2004).

Finally, we briefly describe a range of existing applications including semi-supervised source separation, and discuss the model's possible application to unsupervised source separation.

2 Spectral Deformation Model

Figure 1 shows a narrow band spectrogram representation of a speech signal, where each column depicts the energy content across frequency in a short-time window, or time-frame. The value in each cell is actually the log-magnitude of the short-time Fourier transform; in decibels, $x_t^k = 20log(abs(\sum_{\tau=0}^{N_F-1} w[\tau]x[\tau - t \cdot H]e^{-j2\pi\tau k/N_F}))$, where t is the time-frame index, k indexes the frequency bands, N_F is the size of the discrete Fourier transform, H is the hop between successive time-frames, $w[\tau]$ is the N_F-point short-time window, and $x[\tau]$ is the original time-domain signal. We use 32 ms windows with 16 ms hops. Using the subscript C to designate current and P to indicate previous, the model predicts a patch of N_C time-frequency bins centered at the k^{th} frequency bin of frame t as a "transformation" of a patch of N_P bins around the k^{th} bin of frame $t-1$, i.e.

$$\vec{X}_t^{[k-n_C,k+n_C]} \approx \vec{T}_t^k \cdot \vec{X}_{t-1}^{[k-n_P,k+n_P]} \tag{1}$$

where $n_C = (N_C - 1)/2$, $n_P = (N_P - 1)/2$, and T_t^k is the particular $N_C \times N_P$ transformation matrix employed at that point on the time-frequency plane. We use overlapping patches to enforce transformation consistency, [5](Jojic 2003).

Figure 1 shows an example with $N_C = 3$ and $N_P = 5$ to illustrate the intuition behind this approach. The selected patch in frame t can be seen as a close replica of an upward shift of part of the patch highlighted in frame $t-1$. This "upward" relationship can be captured by a transformation matrix such as the one shown in the figure. The patch in frame $t-1$ is larger than the patch in frame t to permit both upward and downward motions. The generative graphical model for a single layer is depicted in figure 2. Nodes $\mathcal{X} = \{X_1^1, X_1^2, ..., X_t^k, ..., X_T^K\}$ represent all the time-frequency bins in the spectrogram. For now, we consider the continuous nodes $\mathcal{X}$ as observed, although below we will allow some of them to be hidden when analyzing the missing data scenario. Discrete nodes $\mathcal{T} = \{T_1^1, T_1^2, ..., T_t^k, ..., T_T^K\}$ index the set of transformation matrices used to model the dynamics of the signal. Each $N_C \times N_P$ transformation matrix $\vec{T}$ is of the form:

$$\begin{pmatrix} \vec{w} & 0\ 0 \\ 0 & \vec{w} & 0 \\ 0\ 0 & \vec{w} \end{pmatrix} \tag{2}$$

i.e. each of the N_C cells at time t predicted by this matrix is based on the same transformation of cells from $t-1$, translated to retain the same relative relationship. Here, $N_C = 3$ and $\vec{w}$ is a row vector with length $N_W = N_P - 2$; using $\vec{w} = (0\ 0\ 1)$ yields the transformation matrix shown in figure 1. To ensure symmetry along the frequency axis, we constrain N_C, N_P and N_W to be odd. The complete set of $\vec{w}$ vectors include upward/downward shifts by whole bins as well as fractional shifts. An example set, containing

each $\vec{w}$ vector as a row, is:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & .25 & .75 \\ 0 & 0 & 0 & .75 & .25 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & .25 & .75 & 0 \\ . & . & . & . & . \\ .75 & .25 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (3)$$

The length N_W of the transformation vectors defines the supporting coefficients from the previous frame $\vec{X}_{t-1}^{[k-n_W,k+n_W]}$ (where $n_W = (N_W - 1)/2$) that can "explain" X_t^k.

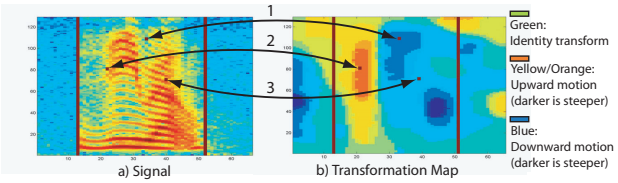


Figure 4: Example transformation map showing corresponding points on original signal.

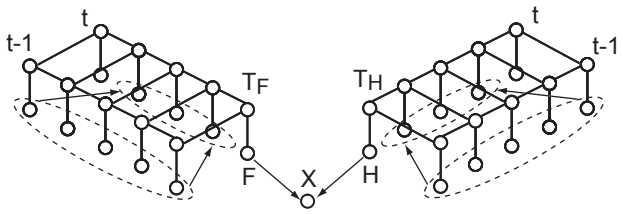


Figure 5: Graphical representation of the two-layer source-filter transformation model.

For harmonic signals in particular, we have found that a model using the above set of $\vec{w}$ vectors with parameters $N_W = 5$, $N_P = 9$ and $N_C = 5$ (which corresponds to a model with a transformation space of 13 different matrices T) is very successful at capturing the self-similarity and dynamics of the harmonic structure.

The transformations set could, of course, be learned, but in view of the results we have obtained with this predefined set, we defer the learning of the set to future work. The results presented in this paper are obtained using the **fixed** set of transformations described by matrix 3.

The clique "local-likelihood" potential between the time-frequency bin X_t^k, its relevant neighbors in frame t, its relevant neighbors in frame $t-1$, and its transformation node T_t^k has the following form:

$$\psi\left(\vec{X}_t^{[k-n_C,k+n_C]}, \vec{X}_{t-1}^{[k-n_P,k+n_P]}, T_t^k\right) = \mathcal{N}\left(\vec{X}_t^{[k-n_C,k+n_C]}; \vec{T}_t^k \vec{X}_{t-1}^{[k-n_P,k+n_P]}, \Sigma^{[k-n_C,k+n_C]}\right) \quad (4)$$

The diagonal matrix $\Sigma^{[k-n_C,k+n_C]}$, which is learned, has different values for each frequency band to account for the variability of noise across frequency bands. For the transformation cliques, the horizontal and vertical transition potentials $\psi_{hor}(T_t^k, T_{t-1}^k)$ and $\psi_{ver}(T_t^k, T_t^{k-1})$, are represented by transition matrices.

For observed nodes $\mathcal{X}$, inference consists in finding probabilities for each transformation index at each time-frequency bin. Exact inference is intractable and is approximated using Loopy Belief Propagation [7, 8] (Yedidia 2001,Weiss 2001) Appendix A gives a quick review of the loopy belief message passing rules, and Appendix B presents the specific update rules for this case.

The transformation map, a graphical representation of the *expected* transformation node across time-frequency, provides an appealing description of the harmonics' dynamics as can be observed in figure 4. In these panels, the links between three specific time-frequency bins and their corresponding transformations on the map are highlighted. Bin 1 is described by a steep downward transformation, while bin 3 also has a downward motion but is described by a less steep transformation, consistent with the dynamics visible in the spectrogram. Bin 2, on other hand, is described by a steep upwards transformation. These maps tend to be robust to noise (see fig 7), making them a valuable representation in their own right.

3 Inferring Missing Data

If a certain region of cells in the spectrogram are missing, like in the case of corrupted data, the corresponding nodes in the model become hidden. This is illustrated in figure 3, where a rectangular region in the center has been removed and tagged as missing. Inference of the missing values is performed again using belief propagation, the update equations are more complex since there is the need to deal with continuous messages, (Appendix C). The posteriors of the hidden continuous nodes are represented using Gaussian distributions, the missing sections on figure 3 part b), are filled in with the means of their inferred posteriors, figure 3 part c), and d). The transformation node posteriors for the missing region are also estimated, in the early stages on the "fill-in" procedure the transformation belief from the "missing" nodes are set to uniform so that the transformation posterior is driven only by the reliable observed neighbors, once the missing values have been filled in with some data, we enable the messages coming from those nodes.

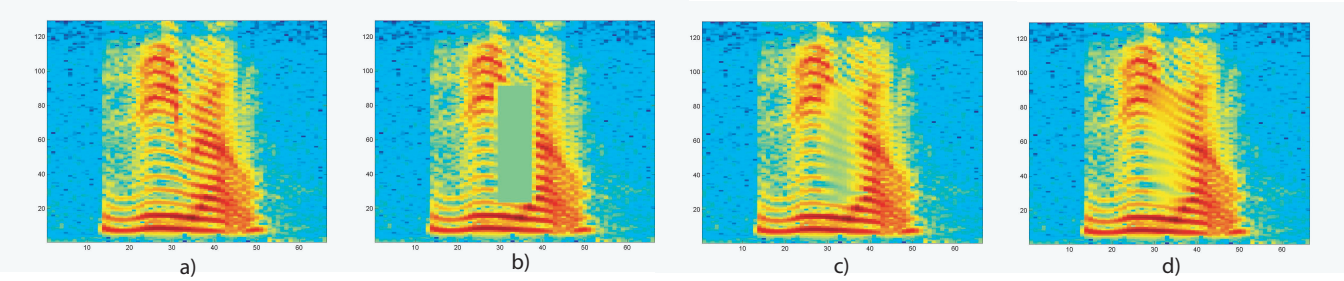


Figure 3: Missing data interpolation example a) Original, b) Incomplete, c) After 10 iterations, d) After 30.

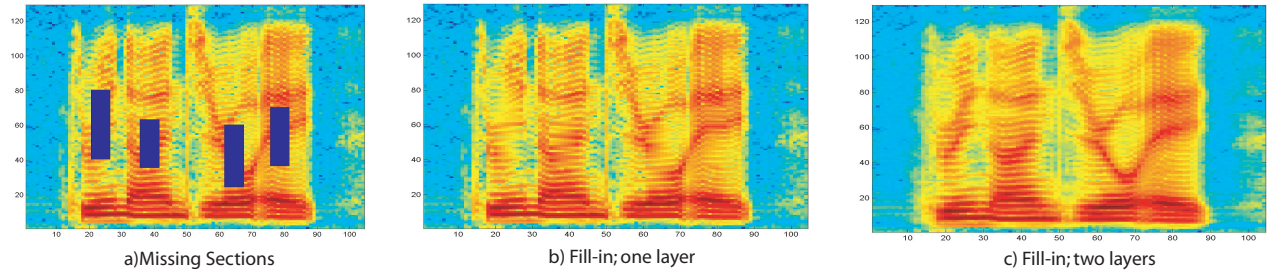


Figure 6: (a) Spectrogram with deleted (missing) regions. (b) Filling in using a single-layer transformation model. (c) Results from the two-layer model.

4 Two Layer Source-Filter Transformations

Many sound sources, including voiced speech, can be successfully regarded as the convolution of a broad-band *source excitation*, such as the pseudo-periodic glottal flow, and a time-varying resonant *filter*, such as the vocal tract, that 'colors' the excitation to produce speech sounds or other distinctions. When the excitation has a spectrum consisting of well-defined harmonics, the overall spectrum is in essence the resonant frequency response sampled at the frequencies of the harmonics, since convolution of the source with the filter in the time domain corresponds to multiplying their spectra in the Fourier domain, or adding in the log-spectral domain. Hence, we model the log-spectra X as the sum of variables F and H, which explicitly model the formants and the harmonics of the speech signal. The source-filter transformation model is based on two additive layers of the deformation model described above, as illustrated in figure 5. Variables F and H in the model are hidden, while, as before, X can be observed or hidden. The symmetry between the two layers is broken by using different parameters in each, chosen to suit the particular dynamics of each component. We use transformations with a larger support in the formant layer ($N_W = 9$) compared to the harmonics layer ($N_W = 5$). Since all harmonics tend to move in the same direction, we enforce smoother transformation maps on the harmonics layer by using potential transition matrices with higher self-loop probabilities. An example of the transformation map for the formant layer is shown in figure 7, which also illustrates how these maps can remain relatively invariant to high levels of signal corruption; belief propagation searches for a consistent dynamic structure within the signal, and since noise is less likely to have a well-organized structure, it is properties of the speech component that are extracted. Inference in this model is more complex, but the actual form of the continuous messages is essentially the same as in the one layer case (Appendix C), with the addition of the potential function relating the signal X_t^k with its transformation components H_t^k and F_t^k at each time-frequency bin:

$$\psi(X_t^k, H_t^k, F_t^k) = \mathcal{N}(X_t^k; H_t^k + F_t^k, \sigma^k) \qquad (5)$$

The first row of figure 10 shows the decomposition of a speech signal into harmonics and formants components, illustrated as the means of the posteriors of the continuous hidden variables in each layer. The decomposition is not perfect, since we separate the components in terms of differences in dynamics; this criteria becomes insufficient when both layers have similar motion. However, separation improves modeling precisely when each component has a different motion, and when the motions coincide, it is not really important in which layer the source is actually captured. Figure 6 a) shows the first spectrogram from figure 10 with deleted regions; notice that the two layers have distinctly different motions. In b) the regions have been filled via inference in a single-layer model; Notice that since the formant motion does not follow the harmonics, the formants are not captured in the reconstruction. In c) the two layers are first decomposed and then each layer is filled in; the figure shows the addition of the filled-in

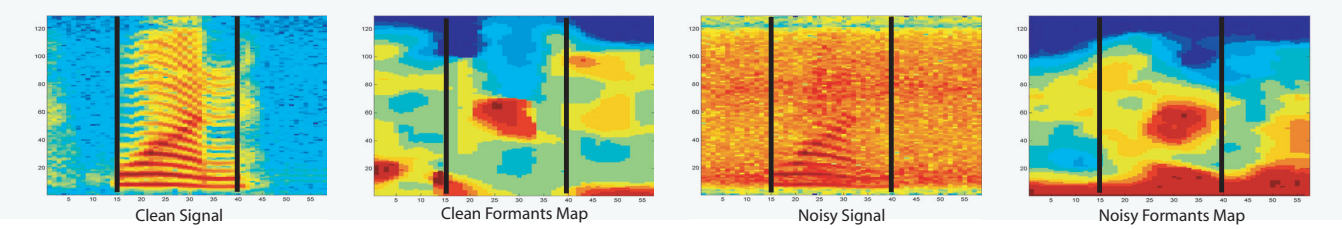


Figure 7: Formant tracking map for clean speech (left panels) and speech in noise (right panels).

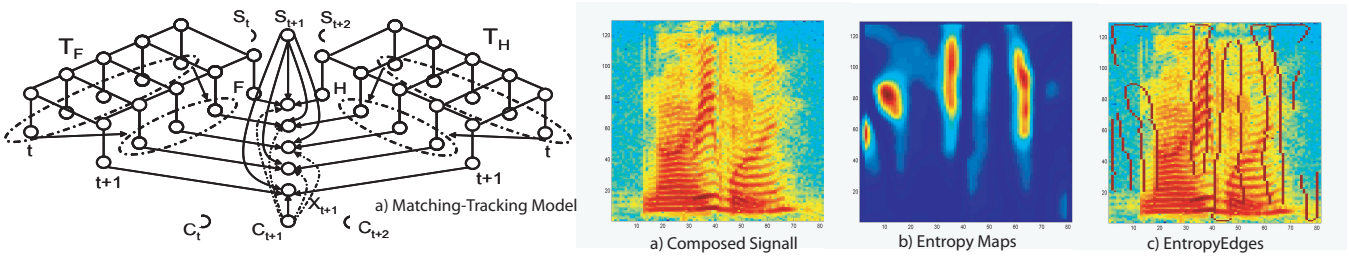


Figure 8: Left: Graphic model of the matching-tracking model; Right: Entropy Map and Entropy Edges

version in each layer.

5 Matching-Tracking Model

Prediction of frames from their context is not always possible such as when there are transitions between silence and speech or transitions between voiced and unvoiced speech, so we need a set of states to represent these unpredictable frames explicitly. We will also need a second "switch" variable that will decide when to "track" (transform) and when to "match" the observation with a state. The first row of figure 8 shows a graphical representation of this model. At each time frame, discrete variables S_t and C_t are connected to all frequency bins in that frame. S_t is a uniformly-weighted Gaussian Mixture Model containing the means and the variances of the states to model. Variable C_t takes two values: When it is equal to 0, the model is in "tracking mode"; a value of 1 designates "matching mode". The potentials between observations x_t^k, harmonics and formants hidden nodes h_t^k and fk_t respectively, and variables S_t and C_t is given by:

$$\psi\left(x_t^k, h_t^k, f_t^k, S_t, C_t = 0\right) = \mathcal{N}\left(x_t^k; h_t^k + f_t^k, \sigma^k\right) \quad (6)$$

$$\psi\left(x_t^k, h_t^k, f_t^k, S_t = j, C_t = 1\right) = \mathcal{N}\left(x_t^k; \mu_j^k, \phi_j^k\right) \quad (7)$$

Inference is done again using loopy belief propagation. Defining ϕ as a diagonal matrix, the M-Step is given by:

$$\begin{aligned} \mu_j &= \frac{\sum_t (Q(S_t = j)Q(C_t = 0)X_t)}{\sum_t (Q(S_t = j)Q(C_t = 0))} \\ \sigma_k &= \frac{\sum_t (Q(C_t = 1)(x_t^k - (f_t^k + h_t^k)))^2}{\sum_t (Q(C_t = 1))} \\ \phi_j &= \frac{\sum_t (Q(S_t = j)Q(C_t = 0)(X_t - \mu_j))^2}{\sum_t (Q(S_t = j)Q(C_t = 0))} \end{aligned} \quad (8)$$

$Q(S_t)$ and $Q(C_t)$ are obtained using the belief propagation rules. $Q(C_t = 0)$ is large if eqn. 6 is larger than eqn. 7. In early iterations when the means are still quite random, eqn. 6 is quite large, making $Q(C_t = 0)$ large with the result that the explicit states are never used.

To prevent this we start the model with large variances ϕ and σ, which will result in non-zero values for $Q(C_t = 1)$, and hence the explicit states will be learned.

As we progress, we start to learn the variances by annealing the thresholds i.e. reducing them at each iteration. We start with a relatively large number of means, but this becomes much smaller once the variances are reduced; the lower-thresholds then control the number of states used in the model. The resulting states typically consist of single frames at discontinuities as intended. Figure 9 a) shows the frames chosen for a short speech segment, (the spectrogram on figure 3.), the signal can be regenerated from the model using the states and both estimated motion fields. The reconstruction is simply another instance of inferring missing values, except the motion fields are not reestimated since we have the true ones. Figure 9 shows several stages of the reconstruction.

6 Applications

We have built an interactive model that implements formant and harmonics tracking, missing data interpolation, formant/harmonics decomposition, and semi-supervised source separation of two speakers. Videos illustrating the use of this demo are available at: `http://www.ee.columbia.edu/~mjr59/def_spec.html`.

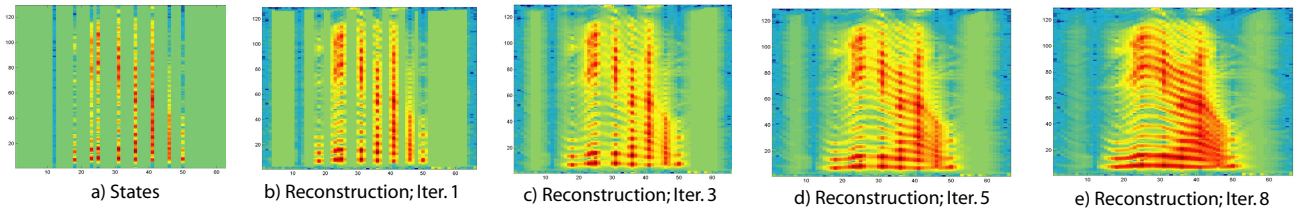


Figure 9: Reconstruction from the matching-tracking representation, starting with just the explicitly-modeled states, then progressively filling in the transformed intermediate states.

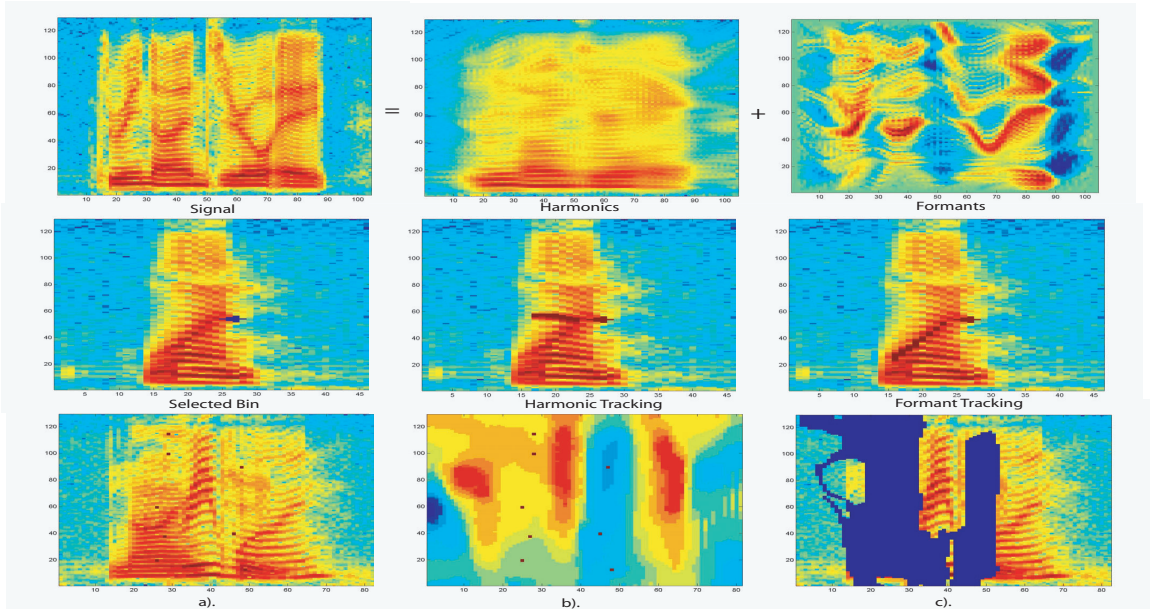


Figure 10: First row: Harmonics/Formants decomposition (posterior distribution means). Row 2: Harmonics/Formants tracking example. The transformation maps on both layers are used to track a given time-frequency bin. Row 3: Semi-supervised Two Speakers Separation. a) The user selects bins on the spectrogram that she believes correspond to one speaker. b) The system finds the corresponding bin on the transformation map. c) The system selects all bins whose transformations match the ones chosen; the remaining bins correspond to the other speaker.

Formants and Harmonics Tracking: Analyzing a signal with the two-layer model permits separate tracking of the harmonic and formant 'ancestors' of any given point. The user clicks on the spectrogram to select a bin, and the system reveals the harmonics and formant "history" of that bin, as illustrated in the second row of figure 10.

Semi-Supervised Source Separation: After modeling the input signal, the user clicks on time-frequency bins that appear to belong to a certain speaker. The demo then masks all neighboring bins with the same value in the transformation map; the remaining unmasked bins should belong to the other speaker. The third row of figure 10 depicts an example with the resultant mask and the "clicks" that generated it. Although far from perfect, the separation is good enough to perceive each speaker in relative isolation.

Missing Data Interpolation and Harmonics/Formants Separation: Examples of these have been shown above.

Features for Speech Recognition: The phonetic distinctions at the basis of speech recognition reflect vocal tract filtering of glottal excitation. In particular, the dynamics of formants (vocal tract resonances) are known to be powerful "information-bearing elements" in speech. We believe the formant transformation maps may be a robust discriminative feature to be use in conjunction with traditional features in speech recognition systems, particularly in noisy conditions; this is future work.

7 Potential Unsupervised Source Separation Applications

The right hand of figure 8 illustrates the *entropy* of the distributions inferred by the system for each transforma-

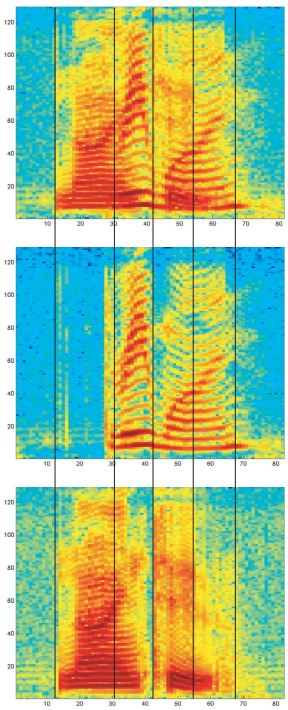


Figure 11: First pane shows the composed spectrogram, second and third spectrograms correspond to the individual sources, vertical lines correspond to the frames learned as states. Notice how the model captures the switches of dominant speaker.

tion variable on a composed signal. The third pane shows 'entropy edges', boundaries of high transformation uncertainty. With some exceptions, these boundaries correspond to transitions between silence and speech, or when occlusion between speakers starts or ends. Similar edges are also found at the transitions between voiced and unvoiced speech. High entropy at these points indicates that the model does not know what to track, and cannot find a good transformation to predict the following frames. These "transition" points are captured by the state variables when the Matching-Tracking model is applied to a composed signal, figure 11, the state nodes normally capture the first frame of the "new dominant" speaker. The source separation problem can be addressed as follows: When multiple speakers are present, each speaker will be modeled in its own layer, further divided into harmonics and formants layers. The idea is to reduce the transformation uncertainty at the onset of occlusions by continuing the tracking of the "old" speaker in one layer at the same time as estimating the initial state of the "new" speaker in another layer – a realization of the "old-plus-new" heuristic from psychoacoustics. This is part of our current research.

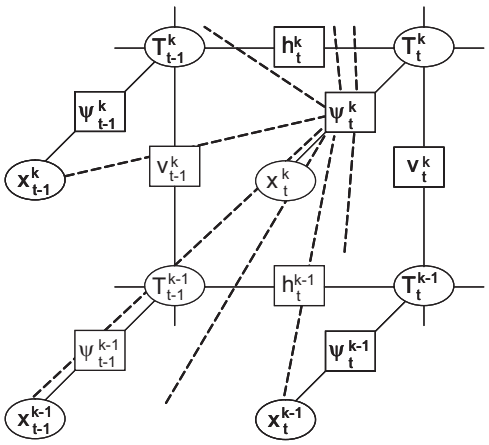


Figure 12: Factor Graph

8 Conclusions

We have presented a harmonic/formant separation and tracking model that effectively identifies the different factors underlying speech signals. We show that this model has a number of useful applications, several of which have already been implemented in a working real-time demo. The model we have proposed in this paper captures the details of a speech signal with only a few parameters, and is a promising candidate for sound separation systems that do not rely on extensive isolated-source training data.

9 Appendices

A: Loopy Belief Propagation

The sum-product algorithm [9](Kschischang 2001) can be used to approximate inference on graphical models with loops. The algorithm update rules applied to the factor graph representation of the model are:
Variable to local function:

$$m_{x\to f}(x) = \prod_{h\in n(x)\setminus f} m_{f\to x}(x) \qquad (9)$$

Local function to variable:

$$m_{f\to x}(x) = \sum_{\sim x} f(X) \prod_{y\in n(f)\setminus x} m_{y\to f}(y) \qquad (10)$$

where $X = n(f)$ is the set of arguments of the function f.

B: Update Rules for the Spectral Deformation Model

Figure 12 depicts a section of the factor graph representation of our model. Function nodes h_t^k and v_t^k represent respectively the potential cliques (transition matrices) $\psi_{hor}(T_t^k, T_{t-1}^k)$ and $\psi_{ver}(T_t^k, T_t^{k-1})$. Function node ψ_t^k, which represents the local likelihood potential defined in eq. 4, is connected to N_C "observation" variables in frame

t ([$x_t^{k-n_C}..x_t^{k+n_C}$], n_C = (N_C - 1)/2) and to N_P "observation" variables in frame $t-1$.

When variables x_t^k are actually observed, only discrete messages between function nodes h_t^k, v_t^k and variable nodes T_t^k are required by the algorithm. Applying recursively the above update rules, we obtain the following forward recursion for the horizontal nodes on the grid:

$$m_{T_t^k \to h_t^k}(T_t^k) = (\sum_{T_{t-1}^k} h_t^k(T_t^k, T_{t-1}^k) m_{T_{t-1}^k \to h_{t-1}^k}(T_{t-1}^k)) \psi(\vec{X}_t^{[k-n_C:k+n_C]}, \vec{X}_{t-1}^{[k-n_P:k+n_P]}, T_t^k) g(T_t^k) \quad (11)$$

where $g(T_t^k) = m_{v_t^k \to T_t^k}(T_t^k) m_{v_t^{k+1} \to T_t^k}(T_t^k)$. A similar backward recursion can also be found. The messages for the vertical chains can be updated through analogous upward/downward recursions.

C: Loopy Belief with Continuous-Valued Messages

The message from function node ψ_t^k to variable x_j^i has the form.

$$m_{\psi_t^k \to x_j^i}(x_j^i) = \int_{\vec{y},\vec{z}} \frac{1}{C} exp^{\frac{1}{2}(\alpha x_j^i - \Gamma\vec{y} + \vec{z})' \Sigma^{-1}_{[r-n_C:r+n_C]} (\alpha x_j^i - \Gamma\vec{y} + \vec{z})} \mathcal{N}(\vec{y}; \mu_y, \Sigma_y) \mathcal{N}(\vec{z}; \mu_z, \Sigma_z) d\vec{y} d\vec{z} \quad (12)$$

Where j is either $t-1$ or t and $i \in [k-n_P, k+n_P]$ if $j = t-1$ or $i \in [k-n_C, k+n_C]$ if $j = t$. Vector $\vec{y}$ is formed by the values on $X_{t-1}^{[r-n_P:r+n_P]}$ other than x_j^i if $j = t-1$ or the whole vector if $j = t$. Vectors $\vec{z}$ and $\vec{X}_t^{[r-N_C:r+N_C]}$ have an analogous relationship. Vector α and matrix Γ come from the most likely (or weighted mean) of the transformation matrix used at T_t^k.

Vectors $\vec{y}$ and $\vec{z}$ are obtained by concatenating individual variables x_r^s. Therefore $\mathcal{N}(\vec{y}; \mu_y, \Sigma_y)$ and $\mathcal{N}(\vec{z}; \mu_z, \Sigma_z)$ should be obtained by completing the square of the multiplication of the gaussian messages from the relevant individual variables x_r^s to the function node ψ_t^k. For simplicity and to speed up the process we approximate them instead by delta functions $\delta(\vec{y} - \mu_y)$ and $\delta(\vec{z} - \mu_z)$, where μ_y and μ_z are obtained as explain below. Then the messages reduce to: $m_{\psi_t^k \to x_j^i}(x_j^i) = \frac{1}{C} exp^{\frac{1}{2}(\alpha x_j^i - \Gamma\mu_y + \mu_z)' \Sigma^{-1} (\alpha x_j^i - \Gamma\mu_y + \mu_z)}$.

The posterior probability of node x_t^k, $q(x_t^k)$, is equal to the multiplication of all its incoming messages. We approximate this multiplication with a Gaussian distribution, $q'(x_t^k) = \mathcal{N}(x_t^k; \mu_{x_t^k}, \phi_{x_t^k})$. Minimizing their KL divergence we find:

$$\mu_{x_t^k} = \frac{\sum_{i=1}^{N_C+N_P} \alpha_i' \Sigma_i^{-1} (\Gamma_i \vec{y}_i - \vec{z}_i)}{\sum_{i=1}^{N_C+N_P} \alpha_i' \Sigma_i^{-1} \alpha_i^{-1}} \quad (13)$$

The values displayed by the missing data application are these mean values. The means of the variable to local function nodes messages, $m_{x_t^k \to \psi_j^i}(x_t^k)$, have the same form as in equation 13, just subtracting the numerator and denominator factor corresponding to the incoming message from the corresponding function. Since we use diagonal variances, parameters μ_y and μ_z in 12 are found by concatenating the means of the relevant messages $m_{x_t^k \to \psi_j^i}(x_t^k)$. When using the two layer model, an extra message comes from the other layer adding extra factors in the numerator and denominator of equation 13.

Acknowledgements

This work was supported by Microsoft Research and by the NSF under grant no. IIS-0238301. We want to thank Sumit Basu for insightful feedback on this research. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References

[1] S. Roweis, "One-microphone source separation", Advances in NIPS, MIT Press, 2000.

[2] J. Bilmes, "Data-driven extensions to HMM statistical dependencies", Proc. ICSLP, 1998.

[3] N. Jojic and B. Frey, "Learning flexible sprites in video layers", Proc. CVPR, 2001.

[4] A. Levin, A. Zomet, and Y. Weiss "Learning to perceive transparency from the statistics of natural scenes", Proc. NIPS, 2002.

[5] N. Jojic, B. Frey, and A. Kannan, "Epitomic Analysis of Appearance and Shape", Proc. ICCV, 2003.

[6] M. Reyes-Gomez, N. Jojic, and D. Ellis, "Towards single-channel unsupervised source separation of speech mixtures:The layered harmonics/formants separation-tracking model", SAPA04. Korea 2004.

[7] J.S. Yedidia, W.T. Freeman, and Y. Weiss, "Understanding Belief Propagation and its Generalizations", Exploring Artificial Intelligence in the New Millennium, Chapter 8.

[8] Y. Weiss and W.T. Freeman, "Correctness of Belief Propagation in Gaussian Graphical Models of Arbitrary Topology", Neural Computation, V13, No 10, pp 2173-2200, 2001.

[9] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor Graphs and the Sum-Product Algorithm", IEEE Transactions on information theory, Vol. 47 No. 2, 2001.

Audio demonstrations at http:// www.comm.utoronto.ca/ rennie/srcsep

Variational Speech Separation of More Sources than Mixtures

Steven J. Rennie, Kannan Achan, Brendan J. Frey, Parham Aarabi
Department of Electrical and Computer Engineering, University of Toronto
rennie@eecg.utoronto.ca

Abstract

We present a novel structured variational inference algorithm for probabilistic speech separation. The algorithm is built upon a new generative probability model of speech production and mixing in the full spectral domain, that utilizes a detailed probability model of speech trained in the magnitude spectral domain, and the position ensemble of the underlying sources as a natural, low-dimensional parameterization of the mixing process. The algorithm is able to produce high quality estimates of the underlying source configurations, even when there are $more$ underlying sources than available microphone recordings. Spectral phase estimates of all underlying speakers are automatically recovered by the algorithm, facilitating the direct transformation of the obtained source estimates into the time domain, to yield speech signals of high perceptual quality.

1 Introduction

The speech separation problem is one that has been very heavily researched, and whose solution under practical conditions still alludes us today. Several existing approaches work well under various problem assumptions – such as negligible or stationary reverberation, instantaneous mixing, or more microphones recordings than speech sources – but break down when these conditions are relaxed.

Two important directions of progress in speech separation research have been the incorporation of detailed information about the nature of speech into the estimation process, and the utilization of multiple signal mixtures (Frey et al. 2001; Bell and Sejnowski 1995). Currently the emphasis of much research is on utilizing these methodologies simultaneously (Attias 2003; A.Acero, Altschuler and Wu 2000; Rennie et al. 2003). Approximate inference techniques have been applied to the problem to facilitate the incorporation of more representative models of speech production and mixing into the estimation process, with success (Attias 2003; Rennie et al. 2003). Spatially selective (e.g. beamforming) algorithms, on the other hand, have demonstrated significant results via the utilization of source position or direction information, despite the fact that the majority of existing techniques do fully decoupled source estimation, and do not incorporate prior information about the nature of speech (Aarabi and Shi 2004; Cohen and Berdugo 2002; Nix, Kleinschmidt and Hohmann 2003).

In this paper, we present a novel structured variational inference algorithm for probabilistic speech separation. The algorithm is built upon a new generative probability model of speech production and mixing in the full spectral domain, that utilizes a detailed probability model of speech trained in the magnitude spectral domain, and the position ensemble of the underlying sources as a natural, low-dimensional parameterization of the mixing process.

For the case where the locations of the underlying speakers are known, the algorithm is able to produce high quality estimates of the underlying source configurations, even when there are $more$ underlying sources than available microphone recordings. When only noisy estimates of the positions of the underlying speakers are available, the algorithm is automatically able to refine the position estimates, improving the achieved separation results substantially. The algorithm also automatically recovers high fidelity estimates of the spectral phase of the underlying speakers, facilitating the direct transformation of the obtained source estimates into the time domain, to yield speech signals of high perceptual quality.

2 The Mixing Process

We model the signal received by microphone m of a collection of microphones M as a scaled, time-delayed com-

bination of all underlying speech sources, and noise:

$$x_m(t) = \sum_S k_{m,s} z_s(t - \tau_{m,s}) + n_m(t) \quad (1)$$

where $\tau_{m,s}$ and $k_{m,s}$ are the time delay and intensity decay associated with the propagation of source signal s to microphone observation m, and n_m represents *all* noise corruption (including transduction noise, other acoustic sources, and reverberation when present).

Both the propagation delay and the intensity decay associated with a given source are a function of the position of the source, $\boldsymbol{\rho}_s$, relative to that of the microphone, $\boldsymbol{\rho}_m$, and the propagation media. Under generally encountered indoor conditions (negligible wind and temperature gradients), the relationship between $\tau_{m,s}$ and $\boldsymbol{\rho}_s$ given $\boldsymbol{\rho}_m$ can be approximated to high fidelity as frequency independent and geometric:

$$\tau_{m,s} = \tau_{m,s}(\boldsymbol{\rho}_s) = \frac{\|\boldsymbol{\rho}_s - \boldsymbol{\rho}_m\|}{v_s} \quad (2)$$

where v_s is the speed of sound in air. Similarly, under generally encountered room conditions the intensity decay associated with atmospheric effects such as wind and temperature gradients, and molecular absorption, are negligible compared to the intensity decay associated with the geometric spread of the acoustic signal from its origin. As such the intensity of the source signal decay is proportional to one over the distance from the source:

$$k_{m,s} = k_{m,s}(\boldsymbol{\rho}_s) = \frac{k_s \cdot g_m}{\|\boldsymbol{\rho}_s - \boldsymbol{\rho}_m\|} \quad (3)$$

where g_m is the gain associated with the mth transducer, and k_s is a generally unknown constant that equalizes the source signals observed at the microphones relative to a chosen reference.

An equivalent representation of the relation (1) in the frequency domain is given by:

$$\begin{bmatrix} \mathbf{x}_{1_\omega} \\ \mathbf{x}_{2_\omega} \\ \vdots \\ \mathbf{x}_{M_\omega} \end{bmatrix} = \mathbf{A}_\omega(\boldsymbol{\rho}) \begin{bmatrix} \mathbf{z}_{1_\omega} \\ \mathbf{z}_{2_\omega} \\ \vdots \\ \mathbf{z}_{S_\omega} \end{bmatrix} + \begin{bmatrix} \mathbf{n}_{1_\omega} \\ \mathbf{n}_{2_\omega} \\ \vdots \\ \mathbf{n}_{M_\omega} \end{bmatrix} \quad (4)$$

where the matrix $\mathbf{A}_w(\boldsymbol{\rho})$ consists of 2×2 blocks $\mathbf{A}_{w_{m,s}}(\boldsymbol{\rho}_s)$ of the form:

$$\mathbf{A}_{w_{m,s}}(\boldsymbol{\rho}_s) = k_{m,s} \begin{bmatrix} \cos\omega\tau_{m,s} & \sin\omega\tau_{m,s} \\ -\sin\omega\tau_{m,s} & \cos\omega\tau_{m,s} \end{bmatrix} \quad (5)$$

and $\mathbf{z}_{s_\omega}$ is the Short-Time Discrete Fourier Transform of the sth (sampled) sound source signal at center frequency ω in vectored form:

$$\mathbf{z}_{s_\omega} = \begin{bmatrix} Re\{\sum_n z_s[n]w[n]e^{-j\frac{2\pi k}{N}n}\} \\ Im\{\sum_n z_s[n]w[n]e^{-j\frac{2\pi k}{N}n}\} \end{bmatrix}, \omega = \frac{k}{N}\omega_s \quad (6)$$

and $\mathbf{x}_{m_\omega}$ is similarly defined. Here $w[n]$ is a (generally non-rectangular) windowing function that is non-zero over N contiguous samples of the sampled source signal $z_s[n] = z_s(nT_s)$ that we wish to generate a spectral representation of, and $w_s = 2\ /T_s$ is the sampling rate, in radians per second.

Applying (4) over segments of length such that the error in the relation due to windowing and the assumption of signal stationarity is minimal (typically 10-20 ms for speech), we have for each segment, given the source position ensemble $\boldsymbol{\rho} = \{\boldsymbol{\rho}_s\}$, a system of $linear$ equations constraining the underlying source signal spectra. Furthermore we have expressed the mixing process in terms of the underlying low dimensional manifold – defined by the positions of the underlying speakers – which relates the observed mixtures to the direct signal component of the speech sources.

3 Modelling Speech in the Full Spectral Domain

When doing speech separation on real microphone recordings in the frequency domain, the mixing process has both amplitude and phase components, and so to incorporate prior information about the nature of speech it is essential to move to the full spectral domain, so that both amplitude and phase corruption can be filtered.

Here the fidelity of the recovered spectral magnitude and phase estimates will be coupled for each source, and across sources: therefore even in cases where we are interested only in recovering the magnitude spectrum of a given speaker (for input to a machine recognition system, for example) phase representation during source inference is critical.

In cases where a time domain estimate of one or more sources is of interest, the spectral phase of the estimate recovered in the frequency domain will greatly affect the perceptual quality of the obtained result. Recent research efforts on the reconstruction of speech given only it's energy spectrum have demonstrated the importance of phase on perceptual quality, and the difficulty of the problem (Achan, Roweis and Frey 2003).

Although it is well known that the spectral phase of speech is coupled across harmonics, this knowledge is difficult to utilize in practice, as frequency sampling complicates the theoretically straightforward relationship. The definition of spectral phase relationships across adjacent analysis frames are similarly complicated by the discretization of frequency. No one has yet identified any utility in the phase of speech for as a feature for sound discrimination or recognition. The magnitude spectrum of speech (or transform of the magnitude spectrum), on the other hand, is established as an excellent feature domain for speech analysis. Speech sounds are characterized by their spectral magnitude pro-

file across frequency, and across time. LPC and Gaussian-based (HMM,Mixture) models are the current representations of choice for capturing these relationships (Rabiner and Juang 1993; Frey et al. 2001; Attias 2003). These observations collectively lead us to seek a probability model of speech in the full spectral domain that incorporates detailed information about the nature of speech (as characterized in the magnitude spectral domain), and is phase-invariant across both frequency and time.

Based on the forgoing discussion then, we define a phase-invariant model of speech in the full spectral domain as follows. We map a learned HMM model of speech in the magnitude spectral domain into the full spectral domain by rotating the (diagonal covariance) Gaussian state emission distributions, at each frequency, at discrete, regular intervals, and introducing phase covariance proportional to the chosen interval size. The result is a generative model of speech in the full spectral domain that is approximately phase-invariant:

$$p(\mathbf{z}_s) = \frac{1}{Z_{\theta_s}} \sum_{\mathbf{c_s},\boldsymbol{\theta}_s} p(c_{s_0}) \prod_{t=0}^{T-1} p(c_{s_{t+1}}|c_{s_t}) \prod_{t=0}^{T} p(\mathbf{z}_{s_t}|c_{s_t},\boldsymbol{\theta}_{s_t}) \quad (7)$$

$$p(\mathbf{z}_{s_t}|c_{s_t},\boldsymbol{\theta}_{s_t}) = N(\mathbf{z}_{s,t};\boldsymbol{\mu}_{c_{s_t},\boldsymbol{\theta}_{s_t}},\boldsymbol{\Sigma}_{c_{s_t},\boldsymbol{\theta}_{s_t}}),$$

$$p(c_{s_{t+1}}|c_{s_t}) = a_{c_{s_{t+1}},c_{s_t}}, p(c_{s_0}) = \quad c_s$$

$$\boldsymbol{\mu}_{c_{s_t},\boldsymbol{\theta}_{s_t}} = R_{\boldsymbol{\theta}_{s_t}} \boldsymbol{\mu}_{c_{s_t}}, \quad \boldsymbol{\Sigma}_{c_{s_t},\boldsymbol{\theta}_{s_t}} = R_{\boldsymbol{\theta}_{s_t}} \boldsymbol{\Sigma}_{c_{s,t}} R^T_{\boldsymbol{\theta}_{s_t}}$$

where the random variables c_{s_t} and $\boldsymbol{\theta}_{s_t}$ represent the underlying state configuration, and the coarse phase of speech source s during time frame t, respectively. $\boldsymbol{\mu}_{c_{s_t}}$ and $\boldsymbol{\Sigma}_{c_{s_t}}$ are the mean and diagonal covariance of the emission distribution of state c_{s_t} for $\boldsymbol{\theta}_{s_t} = \mathbf{0}$, and $R_{\boldsymbol{\theta}_{s,t}}$ is a deterministic rotation matrix given $\boldsymbol{\theta}_{s,t}$.

Figure 1 illustrates how the resulting MOG models of speech in the full spectral domain at each frequency, for a given speech class, are approximately phase invariant as desired.

Note that in the case that the HMM emissions are defined as zero-mean, the model collapses to the more standard model utilized in (Ephraim and Rabiner 1989; Attias 2003), the $\boldsymbol{\theta}_{s_t}$ variable becomes redundant, and phase invariance is automatically achieved. This close relationship allows for the seamless substitution of the more standard model into our source inference algorithm when desired. In particular, we have found that by utilizing the zero-mean source model inference result to initialize source inference under the full probability model (utilizing the non-zero mean source model), the estimation was sped up substantially.

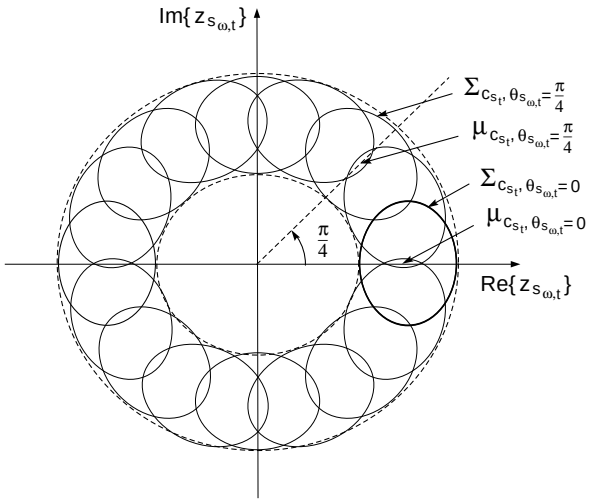


Figure 1: By rotating the source models learned in the magnitude spectral domain at discrete, regular intervals, and introducing phase covariance proportional to the chosen rotation interval size, a phase invariant model of speech in the full spectral domain is obtained.

4 A Generative Model for the Speech Production and Mixing

Based on the foregoing a generative probability model for speech production and mixing over a set of temporally adjacent or overlapping analysis frames T can be written as follows:

$$\begin{aligned} & p(\mathbf{c},\boldsymbol{\theta},\mathbf{z},\boldsymbol{\rho},\mathbf{x}) \\ &= \prod_s p(\mathbf{c}_s) \cdot \prod_{s,t} p(\boldsymbol{\theta}_{s_t}) p(\mathbf{z}_{s_t}|c_{s_t},\boldsymbol{\theta}_{s_t}) \cdot \\ &\qquad \prod_s p(\boldsymbol{\rho}_s) \cdot \prod_t p(\mathbf{x}_t|\mathbf{z}_t,\boldsymbol{\rho}) \\ &= \frac{1}{Z_\theta} \prod_s \quad c_s \prod_{t=0}^{T-1} a_{c_{s_{t+1}},c_{s_t}} \cdot \\ &\quad \prod_{t=0}^{T} N(\mathbf{z}_{s,t}; k_s \boldsymbol{\mu}_{c_{s_t},\boldsymbol{\theta}_{s_t}}, k_s^2 \boldsymbol{\Sigma}_{c_{s_t},\boldsymbol{\theta}_{s_t}}) \cdot \\ &\quad \prod_s N(\boldsymbol{\rho}_s;\boldsymbol{\varrho},\boldsymbol{\varsigma}) \cdot \prod_t \prod_\omega N(\mathbf{x}_{\omega,t}; \mathbf{A}_\omega(\boldsymbol{\rho})\mathbf{z}_{\omega,t}, \boldsymbol{\Psi}_\omega) \end{aligned} \quad (8)$$

where we have modelled noise in the mixing relationship as zero-mean and Gaussian, and treated the positions of the underlying speakers as stationary Gaussian random variables over set of analysis frames (analysis window); an accurate assumption in most settings for analysis windows on the order of 500 ms. Note that the scale equalization parameters k_s have been moved into the definition of source models since they are independent of the microphones. The mixing matrix retains its dependence on scale as a function of source position.

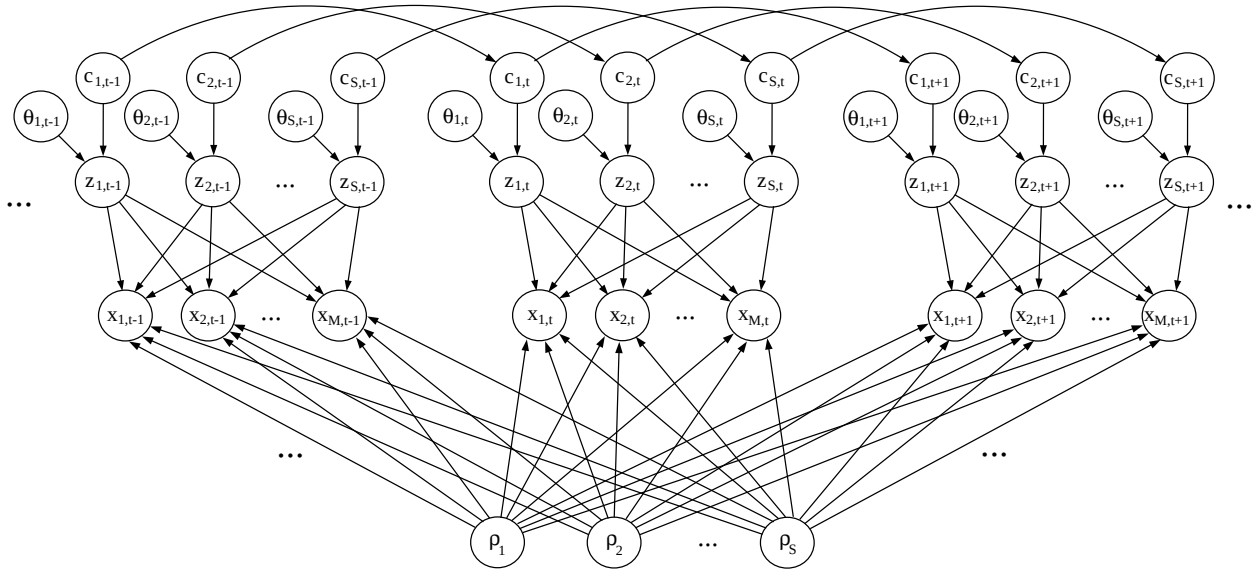


Figure 2: A Bayes net depicting the dependencies that exist between random variables of the speech production and mixing process.

Under this description the generation process for each analysis frame proceeds as follows:

> A speech sound is emitted from each speaker in accordance with the conditional prior $p(c_{s_{t+1}}|c_{s_t}) = a_{c_{s_{t+1}},c_{s_t}}$.
>
> The coarse phase of each speaker at each frequency is uniformly generated from the domain of $\boldsymbol{\theta}_{s_t}$.
>
> Given c_{s_t} and $\boldsymbol{\theta}_{s,t}$, and instance of the speech sound is generated from the distribution of the specified speech cluster for all speakers.
>
> A position ensemble is sampled from the distribution of $\boldsymbol{\rho}$.
>
> Given $\boldsymbol{\rho}$ and $\mathbf{z}_t$, the microphone observations are generated according to $p(\mathbf{x}_t|\mathbf{z}_t, \boldsymbol{\rho}) = \prod_t \prod_\omega N(\mathbf{x}_{\omega,t}; \mathbf{A}_\omega(\boldsymbol{\rho})\mathbf{z}_{\omega,t}, \boldsymbol{\Psi}_\omega)$.

Figure 2 depicts a Bayes net of the generative model presented above.

5 Source Inference

Given our generative probabilistic description of speech production and mixing, the problem of estimating the configuration of the underlying sources over an analysis window given observed microphone mixtures becomes one of simultaneous probabilistic learning and inference, as generally both the configuration of the underlying sources and the parameters of the model $\{\boldsymbol{\Psi}, \mathbf{k}, \boldsymbol{\varrho}, \boldsymbol{\varsigma}\}$ will be unknown. Because the decision about the configuration of the underlying sources is fully coupled by the observed microphone data, even when the positions of the underlying sources are known, exact inference is exponential in the representation complexity of the underlying speech model, and hence generally intractable to compute. However the relationship between the observed mixtures and the underlying sources given the source positions constructed in Section 2 is linear, and the source model defined in Section 3 is built upon Gaussian basis functions, and so the system is conditionally amenable to variational approximate inference techniques (Jordan et al. 1999).

In general however, the position of the underlying sources will not be known, and so a posterior distribution over the source positions must simultaneously estimated. Unfortunately the entries of the mixing matrix are non-linear in the time delays defined by the source positions, and the delays themselves are a non-linear function of the source positions $\boldsymbol{\rho}$, and so density estimation and propagation through these relationships is difficult (and fully coupled) problem, not amenable to analytic approaches.

Here we will concentrate on the case when rough estimates of the underlying source positions are available, and collapse the position ensemble distribution estimation problem onto a point, making it a parameter to be refined during source inference. In making this assumption we remind the reader that source localization in of itself is a very difficult problem, *with today's best acoustic techniques generally requiring many more microphones than sources to achieve position estimates of fidelity* (DiBiase, Silverman

and Brandstein 2001; Aarabi 2003). Note however, that it is the utilization of a naturally existing parameter (the source locations) in defining the mixing process that makes the requirement that some information about the parameter be available plausible.

We achieve simultaneous learning of the unknown parameters of the model and inference of the configuration of the underlying sources by iterating between inferring a structured variational approximation to the posterior distribution of the underlying sources given the current model parameters to define an approximate E-Step and obtain a lower bound the data likelihood, and maximizing the bound with respect to the model parameters $\{\boldsymbol{\rho}, \mathbf{k}, \boldsymbol{\Psi}\}$ to define the M-Step of our Expectation-Maximization algorithm for inferring the configuration of the underlying sources.

E-Step: We define the form of the variational surrogate as:

$$\begin{aligned} & q(\mathbf{z}, \boldsymbol{\theta}, \mathbf{c}) \\ & = \prod_s q(c_{s_0}) \prod_t q(c_{s_{t+1}} | c_{s_t}) \cdot \prod_{s,t,\omega} q(\theta_{s_{\omega,t}}) \cdot \prod_{\omega,t} q(\mathbf{z}_{\omega,t}) \\ & = \prod_s \chi_{s_0} \prod_t \chi_{s_{t+1,t}} \prod_{s,t,\omega} \boldsymbol{\gamma}_{\theta_{s_{\omega,t}}} \prod_{t,\omega} N(\mathbf{z}_{\omega,t}, \boldsymbol{\eta}_{\omega,t}, \boldsymbol{\Omega}_{\omega,t}) \end{aligned} \quad (9)$$

where $\{\boldsymbol{\chi}, \boldsymbol{\gamma}, \boldsymbol{\eta}, \boldsymbol{\Omega}\}$ are the variational parameters to be found so that q best approximates the true posterior of the hidden random variables under our speech separation model. To identify q we minimize the Kullback-Leibler (KL) divergence of $p(\mathbf{z}, \boldsymbol{\theta}, \mathbf{c}, \mathbf{x})$ from $q(\mathbf{z}, \boldsymbol{\theta}, \mathbf{c})$. Exploiting the conditional independencies, conditional linearity, and Gaussian decomposition of the underlying model p given the model parameters, and the chosen form of the variational surrogate, we arrive at the set of coupled fixed point equations for the variational parameters, given in appendix A, that may be iterated to identify q. The computational complexity of the inference algorithm is linear (as opposed to exponential) in the representation complexity of the utilized speech model.

M Step: The update for the source positions $\boldsymbol{\rho}$ is obtained by solving:

$$[\{\partial L / \partial \boldsymbol{\rho}_{s_{x_i}}\} = \mathbf{0} \quad (10)$$

The form of $\partial L / \partial \boldsymbol{\rho}_{s_{x_i}}$ is given in the appendix. The closed form updates for k_s and $\boldsymbol{\Psi}_\omega$ can also be found in the appendix.

6 Results

A database of dictated speech, consisting of 18 minutes of data (3 mins. x 6 female speakers) from the Wall Street Journal database (WSJ) was used to train a 128-component, speaker independent, diagonal covariance Gaussian emission HMM model of speech in the magnitude spectral domain. This model was used to define the (common) source prior in the full spectral domain that was utilized in all our experiments, by isotropically expanding the learned covariances, and rotating the model at intervals of $/32$ (as described in section 3). A 128-component zero-mean, speaker independent, diagonal covariance Gaussian emission HMM model was also trained in the magnitude spectral domain, and mapped directly into the complex domain. For both models, several training trials, (100 EM iterations each) were performed, and the model that maximized the probability of a 12 minute validation database (defined analogously to how the training set was defined) was selected.

A test database of 1 minute of WSJ speech data from each speaker in the training database was used to define the speech sources for all test scenarios presented. Simulated microphone recording were generated via the standard image method (Allen and Berkley 1979), with additional 20 dB Gaussian noise corruption. All simulated scenarios were set in a 7 by 6 by 2.5 m room, with all source and microphone heights set at 1.5m. The horizontal coordinates of the sources and microphones are given in Appendix B. In all the forthcoming results, a non-overlapping, 20 frame analysis window ($T = 20$) was employed, with the 0-4kHz region of the half overlapped, hanning-windowed FFTs of the data (16ms segments) defining each processing frame.

To speed up source inference, for all test scenarios the zero-mean speech model-based version of our speech separation algorithm was first run until convergence, and the inference result was then used to seed our full speech separation algorithm, which was run for an additional 10 EM iterations to yield final source estimates. It worthy of note that in all (non-reverberative) test scenarios, the additional iterations with the non-zero mean source model resulted in substantial (5% to 15%) increases in SNR gain.

Figure 3 depicts spectrograms of a typical microphone recording, and typical separation results achieved for the case of zero reverberation, known source position information, 6 underlying speech sources, and only 4 available microphone observations (Figure 4 depicts the spatial setup of this test scenario). The separation result achieved via norm-constrained inversion of the data likelihood (a beamformer utilizing all source position information):

$$\mathbf{z}^*_{\omega,t_{nc}} = (\mathbf{A}^T_\omega \mathbf{A}_\omega + 0.1\mathbf{I})^{-1} \mathbf{A}^T_\omega \mathbf{x}_{\omega,t}, \; all \; \omega, t \quad (11)$$

are included for comparative purposes. Looking at the results, we can see that our variational inference algorithm is able to yield a dramatic improvement over the norm-constrained inversion based estimate, and recovers a high fidelity estimate of the underlying source, despite the fact that there are two more sources than microphones, and the sources have strongly overlapping spectral-temporal feature content.

Table 1 summarizes the SNR gain results obtained (relative to taking a microphone reading as the source estimates) for

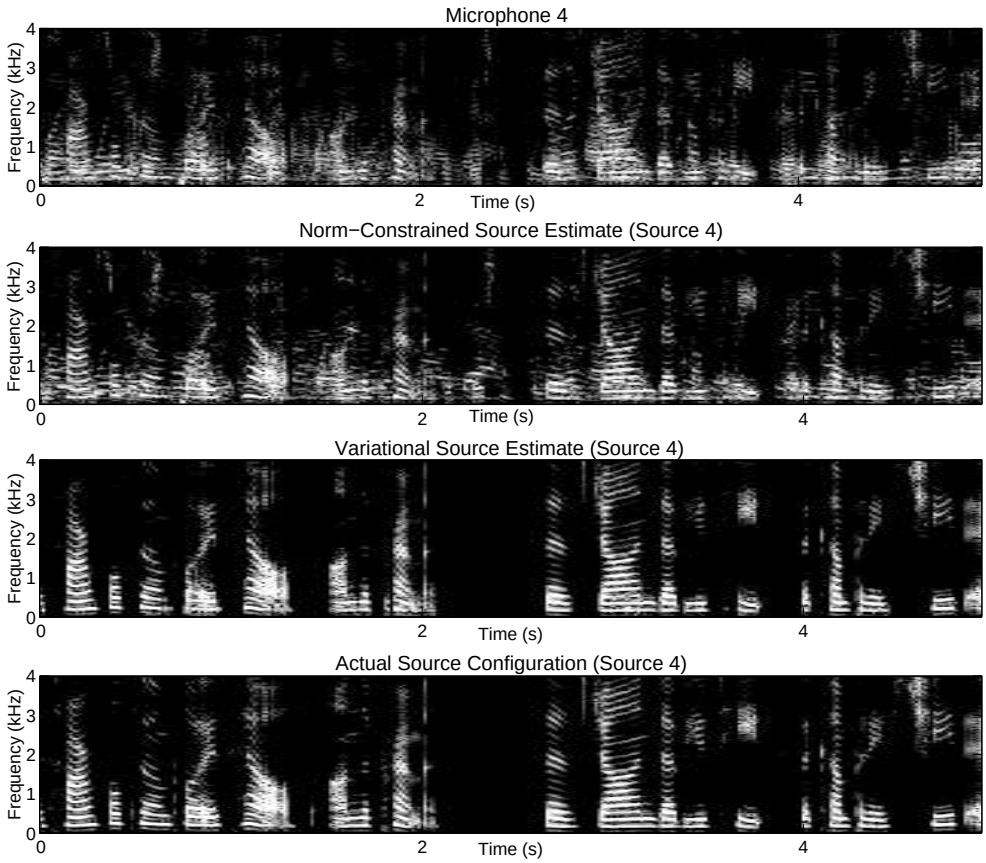


Figure 3: Our algorithm is capable of producing high quality estimates of the magnitude spectra of the underlying sources even when there are more underlying sources than available microphone observations. The obtained SNR Gain over taking a microphone observation as our estimate, and the norm-constrained estimate (11) in this frame is 14.5 and 10.4 dB, respectively.

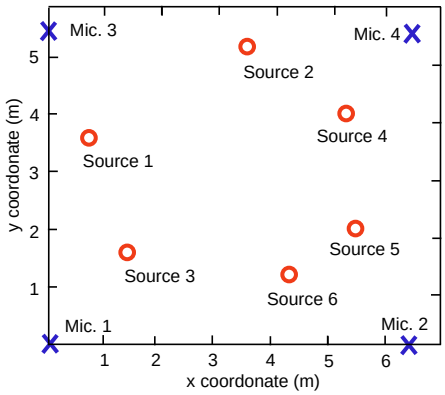


Figure 4: Physical setup of the 4 microphone, 6 speech source test scenario

the case of known source position information, for various source to microphone count combinations, and reverberation times. The average gain of applying the norm-constrained data inversion (beamforming) estimate (11) has also been included in brackets for comparative purposes.

For non-reverberative mixing, our variational algorithm is able to greatly improve upon the beamforming estimate (11), and yields high SNR gain results, even when there are more sources than microphones. The algorithm is automatically able to recover high fidelity estimates of the spectral phase of all sources, to facilitate the direct transformation of the obtained estimates into the time domain, to yield speech signals of high perceptual quality. Audio demonstrations can be listened to at www.comm.utoronto.ca/˜rennie/srcsep.

Table 1: Source vector SNR gain performance of our algorithm as a function of microphone noise corruption level and the number sources and microphones. All gains are calculated in the time domain, and reported in decibels.

Number of Sources	Number of Microphones	Reverberation Time (s) 0	0.05
4	4	23.6 (5.3)	0.3 (1.0)
5	4	18.0 (4.77)	0.3 (1.4)
6	4	14.5 (4.37)	0.4 (0.9)
4	2	5.3 (2.0)	0.9 (1.1)
3	2	9.6 (2.9)	1.3 (1.4)

It is difficult to directly compare our results to existing work. The best performing spatial filtering algorithms that

we are aware of are the aggressive beamforming techniques (Cohen and Berdugo 2002; Aarabi and Shi 2004), which have demonstrated SNR gains in the range of 10 dB at sub-zero dB SNRs. These techniques perform decoupled source inference, and do not incorporate prior information about the nature of speech into the estimation process. Here to no surprise, we have demonstrated much higher fidelity results, by addressing the shortcomings of these algorithms.

Perhaps the most advanced information processing algorithms for speech separation we are aware of are those presented in (Attias 2003). SNR gain results of 3.7 dB and 4.4 dB are reported for the case of 5 sources and 5 microphones and 10 dB microphone noise corruption, and 3 microphones and 2 sources and 10 dB microphone noise corruption, respectively. In this case, however, no knowledge of the spatial locations of the underlying sources was utilized.

In (Nix, Kleinschmidt and Hohmann 2003) a particle filtering algorithm for simultaneous source separation and source direction estimation is presented. Excellent source direction estimation results are presented, but source separation performance results are omitted.

Looking now at our results for reverberant mixing, we can see that in sharp contrast to the non-reverberant mixing results, the algorithm performed very poorly. In (Attias 2003) for example, source vector gains of 7+ dB are reported for more serious reverberative conditions than tested here. We expected the spatial selectivity inherent to the problem formulation to be able to combat some reverberation. Further analysis revealed, however, that the likelihood associated with a given source under the model was only discriminative against sounds immediately around the other sources. The results give us a renewed interest in the operation of the aggressive beamforming techniques (Cohen and Berdugo 2002; Aarabi and Shi 2004) which are highly spatially discriminative.

7 Concluding Remarks

In this paper, a novel structured variational learning and inference algorithm for probabilistic speech separation, built upon a new generative probability model of speech production and mixing, was presented. For the case of multi-path free mixing, and known source position information, excellent separation results were demonstrated. Even in scenarios where there were more sources than microphone observations, the algorithm has demonstrated the ability to automatically recover high quality estimates of the magnitude and phase spectrum of all underlying sources, yielding time domain source estimates of high perceptual quality. We are currently investigating the performance of the algorithm when only noisy position estimates are available. Preliminary results have indicated that when the available source position estimates are within 0.25 meters of the their true values, our algorithm is able to consistently refine the source position estimates; a capability that has yielded a SNR gain performance increase (over assuming the noisy position estimates are correct) consistently over 5 dB and often exceeding 10 dB. More generally the problem of simultaneous source localization and separation constitutes a difficult and open problem; extensions to the presented model may be able to break ground.

We are also interested in pursuing further the presented model of speech in the full spectral domain. Relationships between frequency harmonics, though difficult to utilize in discrete fourier domains, could potentially be exploited by dynamically tuning the analysis frame length to place the pitch period and associated harmonics at the sampled frequencies.

Perhaps the most interesting, and most challenging direction of future work in this research area however, is to investigate new ways of dealing with reverberation.

References

A.Acero, Altschuler, S., and Wu, L. 2000. Speech/noise separation using two microphones and a vq model of speech signals. In *Proceedings of the Internatirional Conference on Spoken Language Processing*.

Aarabi, P. 2003. The fusion of distributed microphone arrays for sound localization. *EURASIP Journal of Applied Signal Processing (Special Issue on Sensor Networks)*, No. 4:338:347.

Aarabi, P. and Shi, G. 2004. Phase-based dual-microphone robust speech enhancement. *IEEE Transactions on Systems, Man, and Cybernetics Part B*, 34.

Achan, K., Roweis, S., and Frey, B. 2003. Probabilistic inference of speech signals from phaseless spectrograms. In *Neural Information Processing Systems 16*.

Allen, J. and Berkley, D. 1979. Image method for efficiently simulating small room acoustics. *JASA*, 65(4):943:950.

Attias, H. 2003. New em algorithms for source separation and deconvolution. In *Proceedings of the IEEE 2003 International Conference on Acoustics, Speech, and Signal Processing*.

Bell, A. and Sejnowski, T. 1995. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159.

Cohen, I. and Berdugo, B. 2002. Microphone array postfiltering for nonstationary noise suppression. In *ICASSP*.

DiBiase, J., Silverman, H., and Brandstein, M. 2001. Robust localization in reverberant rooms. *M.S. Brandstein and D.B. Ward (eds.), Microphone Arrays:Signal Processing Techniques and Applications*.

Ephraim, Y. and Rabiner, L. 1989. A minimum discrimination information approach for hidden markov modeling. *IEEE Transactions on Information Theory*, 35:1001–1013.

Frey, B., Kristjansson, T., Deng, L., and Acero, A. 2001. Learning dynamic noise models from noisy speech for robust speech recognition. In *Proceedings of the 2001 Neural Information Processing Systems (NIPS)*.

Jordan, M. I., Ghahramani, Z., Jaakkola, T., and Saul, L. K. 1999. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

Nix, J., Kleinschmidt, M., and Hohmann, V. 2003. Computational auditory scene analysis by using statistics of high-dimensional speech dynamics and sound source direction. In *EUROSPEECH*.

Rabiner, L. and Juang, B. 1993. *Fundamentals of Speech Recognition*. Prentice-Hall, New Jersey.

Rennie, S., Aarabi, P., Kristjansson, T., Frey, B., and Achan, K. 2003. Robust variational speech separation using fewer microphones than speakers. In *Proceedings of the 2003 IEEE Conference on Acoustics, Speech, and Signal Processing*.

Appendix A: E and M Step Updates

$$\chi_{c_{s_t},c_{s_{t-1}}} \propto a_{c_{s_t},c_{s_{t-1}}} e^{\lambda_{c_{s_t},c_{s_{t-1}}}} \tag{12}$$

$$\lambda_{c_{s_t},c_{s_{t-1}}} = -\frac{1}{2}\log|\boldsymbol{\Sigma}_{c_s}| +$$
$$\sum_{\omega}\sum_{\boldsymbol{\theta}_{s_{\omega,t}}}\{\boldsymbol{\gamma}_{\boldsymbol{\theta}_{s_{\omega,t}}}(\boldsymbol{\mu}_{c_s,\boldsymbol{\theta}_{s_{\omega,t}}}-\boldsymbol{\eta}_{\omega,t})^T\boldsymbol{\Sigma}^{-1}_{c_s,\boldsymbol{\theta}_{s_\omega}}(\boldsymbol{\mu}_{c_s,\boldsymbol{\theta}_{s_{\omega,t}}}-\boldsymbol{\eta}_{\omega,t})+$$
$$\mathrm{Tr}[\boldsymbol{\Sigma}^{-1}_{c_s,\boldsymbol{\theta}_{s_{\omega,t}}}\boldsymbol{\Omega}_{s_{\omega,t}}]\} - D(a_{c_{s_{t+1}},c_{s_t}}\|\chi_{c_{s_{t+1}},c_{s_t}}) + \sum_{c_{s_{t+1}}}\lambda_{c_{s_{t+1}},c_{s_t}}$$

$$\boldsymbol{\gamma}_{\boldsymbol{\theta}_{s_{\omega,t}}} \propto e^{\kappa_{\boldsymbol{\theta}_{s_{\omega,t}}}} \tag{13}$$

$$\kappa_{\boldsymbol{\theta}_{s_{\omega,t}}} = -\frac{1}{2}\sum_{c_{s_t}}\boldsymbol{\chi}_{c_{s_t}}\{(\boldsymbol{\mu}_{c_s,\boldsymbol{\theta}_{s_{\omega,t}}}-\boldsymbol{\eta}_{\omega,t})^T\cdot$$
$$\boldsymbol{\Sigma}^{-1}_{c_s,t,\boldsymbol{\theta}_{s_{\omega,t}}}(\boldsymbol{\mu}_{c_s,\boldsymbol{\theta}_{s_{\omega,t}}}-\boldsymbol{\eta}_{\omega,t})$$

$$\boldsymbol{\eta}_{\omega,t} = \boldsymbol{\Omega}_{\omega,t}(\mathbf{A}^T_\omega\boldsymbol{\Psi}^{-1}_\omega\mathbf{x}_{\omega,t}+\boldsymbol{\zeta}_{\omega,t}) \tag{14}$$

$$\boldsymbol{\Omega}_{\omega,t} = (\mathbf{A}^T_\omega\boldsymbol{\Psi}^{-1}_\omega\mathbf{A}_\omega+\boldsymbol{\Phi}_{\omega,t})^{-1} \tag{15}$$

$$\boldsymbol{\Phi}_{\omega,t} = diag[\sum_{c_{1_t}}\boldsymbol{\chi}_{\mathbf{c_{1_t}}}\sum_{\boldsymbol{\theta}_{1_{\omega,t}}}\gamma_{1_{\omega,t},\boldsymbol{\theta}_{1_{\omega,t}}}\boldsymbol{\Sigma}^{-1}_{c_{1_t},\boldsymbol{\theta}_{1_{\omega,t}}},\ldots,$$
$$\sum_{c_{S_t}}\boldsymbol{\chi}_{\mathbf{c_{S_t}}}\sum_{\boldsymbol{\theta}_{S_{\omega,t}}}\gamma_{S_{\omega,t},\boldsymbol{\theta}_{2_{\omega,t}}}\boldsymbol{\Sigma}^{-1}_{c_{S_t},\boldsymbol{\theta}_{S_{\omega,t}}}]$$

$$\boldsymbol{\zeta}_{\omega_t} = [\sum_{c_{1_t}}\boldsymbol{\chi}_{\mathbf{c_{1_t}}}\sum_{\boldsymbol{\theta}_{1_{\omega,t}}}\boldsymbol{\gamma}_{\boldsymbol{\theta}_{1_{\omega,t}}}\boldsymbol{\Sigma}^{-1}_{c_{1_t},\boldsymbol{\theta}_{1_{\omega,t}}}\boldsymbol{\mu}_{c_{1_t},\boldsymbol{\theta}_{1_{\omega,t}}},\ldots,$$
$$\sum_{c_{S_t}}\boldsymbol{\chi}_{\mathbf{c_{S_t}}}\sum_{\boldsymbol{\theta}_{S_{\omega,t}}}\boldsymbol{\gamma}_{S_{\omega,t},\boldsymbol{\theta}_{2_{\omega,t}}}\boldsymbol{\Sigma}^{-1}_{c_{S_t},\boldsymbol{\theta}_{S_{\omega,t}}}\boldsymbol{\mu}_{c_{S_t},\boldsymbol{\theta}_{S_{\omega,t}}}]$$

$$\partial L/\partial\boldsymbol{\rho}_{s_{x_i}} = \sum_w\sum_t Tr[(\boldsymbol{\Psi}^{-1}_\omega(\mathbf{A}_\omega\boldsymbol{\eta}_{\omega,t}-\mathbf{x}_{\omega,t})\boldsymbol{\eta}^T_{\omega,t}+$$
$$\boldsymbol{\Psi}^{-1}_\omega\mathbf{A}_\omega\boldsymbol{\Omega}^T_{\omega,t})(\partial\mathbf{A}_\omega/\partial\boldsymbol{\rho}_{s_{x_i}})^T] \tag{16}$$

$$\boldsymbol{\Psi}_\omega = \sum_T(\mathbf{x}_\omega-\mathbf{A}_\omega\boldsymbol{\eta}_{\omega,t})(\mathbf{x}_\omega-\mathbf{A}_\omega\boldsymbol{\eta}_{\omega,t})^T \tag{17}$$

$$k_s = \frac{-b_s^2+\sqrt[2]{b_s^2-4a_sc_s}}{2a_s} \tag{18}$$

$$a_s = dim(\mathbf{z}_s)T$$

$$b_s = \sum_{t,c_{s_t}}\chi_{c_{s_t}}\sum_{\omega,\boldsymbol{\theta}_{s_{\omega,t}}}\gamma_{\boldsymbol{\theta}_{s_{\omega,t}}}\boldsymbol{\eta}^T_{\omega,t}\boldsymbol{\Sigma}^{-1}_{c_s,\boldsymbol{\theta}_{s_{\omega,t}}}\boldsymbol{\mu}_{c_s,\boldsymbol{\theta}_{s_{\omega,t}}}$$

$$c_s = -\sum_{t,c_{s_t}}\chi_{c_{s_t}}\sum_{\omega,\boldsymbol{\theta}_{s_{\omega,t}}}\gamma_{\boldsymbol{\theta}_{s_{\omega,t}}}(\boldsymbol{\eta}^T_{\omega,t}\boldsymbol{\Sigma}^{-1}_{c_s,\boldsymbol{\theta}_{s_\omega}}\boldsymbol{\eta}_{\omega,t}+Tr[\boldsymbol{\Sigma}^{-1}_{c_s,\boldsymbol{\theta}_{s_{\omega,t}}}\boldsymbol{\Omega}_{\omega,t}])$$

Appendix B: Test Scenario Details

XSYM - X source, Y microphone test scenario
SP - Source positions
MP - Microphone positions

4S4M:
SP = (0.7,3.6),(3.5, 5.3),(2.8,1.8),(5.3,3.0)
MP = {(0,0),(6.5, 0),(0 5.5),(6.5,5.5)}

5S4M:
SP = {(0.7,3.6),(3.5,5.3),(1.4,1.8),(5.3,3.0),(4.2,1.2)}
MP = {(0,0),(6.5, 0),(0 5.5),(6.5,5.5)}

6S4M:
SP = {(0.7,3.6),(3.5,5.3),(1.4,1.8),(5.3,4.0),(5.6,2.1),(4.2,1.2)}
MP = {(0,0),(6.5, 0),(0 5.5),(6.5,5.5)}

3S2M:
SP = {(0.5,1.5),(3.0,3.0),(6.3,1.2)}
MP = {(1.5,0),(3, 0)}

4S2M:
SP = {(0.5,1.5),(1.4,3.6),(4.6,3.0),(6.3,1.2)}
MP = {(1.5,0),(3, 0)}

Learning Bayesian Network Models from Incomplete Data using Importance Sampling

Carsten Riggelsen and Ad Feelders
Institute of Information & Computing Sciences
Utrecht University
P.O. Box 80098, 3508TB Utrecht
The Netherlands

Abstract

We propose a Bayesian approach to learning Bayesian network models from incomplete data. The objective is to obtain the posterior distribution of models, given the observed part of the data. We describe a new algorithm, called eMC^4, to simulate draws from this posterior distribution. One of the new ideas in our algorithm is to use importance sampling to approximate the posterior distribution of models given the observed data and the current imputation model. The importance sampler is constructed by defining an approximate predictive distribution for the unobserved part of the data. In this way existing (heuristic) imputation methods can be used that don't require exact inference for generating imputations.

We illustrate eMC^4 by its application to modeling the risk factors of coronary heart disease. In the experiments we consider different missing data mechanisms and different fractions of missing data.

1 Introduction

Bayesian networks are probabilistic models that can represent complex interrelationships between random variables. It is an intuitively appealing formalism for reasoning with probabilities that can be employed for diagnosis and prediction purposes. Furthermore, learning Bayesian networks from data may provide valuable insight into the (in)dependences between the variables.

In the last decade, learning Bayesian networks from data has received considerable attention in the research community. Most learning algorithms work under the assumption that *complete* data is available. In practical learning problems one frequently has to deal with missing values however. The presence of incomplete data leads to analytical intractability and high computational complexity compared to the complete data case. It is very tempting to "make the problem go away" either by deleting observations with missing values or using ad-hoc methods to fill in (impute) the missing data. Such procedures may however lead to biased results, and, in case of imputing a single value for the missing data, to an overconfidence in the results of the analysis.

We avoid such ad-hoc approaches and use a method that takes *all* observed data into account, and correctly reflects the increased uncertainty due to missing data. We do assume however that the missing data mechanism is *ignorable* as defined by Little and Rubin (1987). Essentially this means that the probability that some component is missing may depend on observed components, but not on unobserved components.

Our approach is Bayesian in the sense that we are not aiming for a single best model, but want to obtain (draws from) a posterior distribution over possible models. We show how to perform model averaging over Bayesian network models, or alternatively, how to get a range of good models, when we have incomplete data. We develop a method that can handle a broad range of imputation methods without violating the validity of the models returned. Our approach is not restricted to any imputation technique in particular, and therefore allows for imputation methods that do not require expensive inference in a Bayesian network.

This paper is organised as follows. In section 2 we briefly review previous research in this area and show how our work fits in. In section 3 we describe model learning from complete data. In sections 4 and 5 we introduce a new algorithm, called eMC^4, for Bayesian network model learning from incomplete data. We performed a number of experiments to test eMC^4 using

real life data. The results of those experiments are reported in section 6. Finally, we summarize our work and draw conclusions.

2 Previous research

Here we briefly review relevant literature on learning Bayesian networks from incomplete data. Two popular iterative approaches for learning parameters are Expectation-Maximization (EM) by Dempster et al. (1977) and a simulation based Gibbs sampler (Geman and Geman, 1984) called Data Augmentation (DA) introduced by Tanner and Wong (1987). For Bayesian networks EM was studied by Lauritzen (1995). The Expectation step (E-step) involves the performance of inference in order to obtain sufficient statistics. The E-step is followed by a Maximization step (M-step) in which the Maximum Likelihood (ML) estimates are computed from the sufficient statistics. These two steps are iterated until the parameter estimates converge.

Data Augmentation (DA) is quite similar but is nondeterministic. Instead of calculating expected statistics, a value is drawn from a predictive distribution and imputed. Similarly, instead of calculating the ML estimates, one draws from the posterior distribution on the parameter space (conditioned on the sufficient statistics of the most recent imputed data set). Based on Markov chain Monte Carlo theory this will eventually return realizations from the posterior parameter distribution. There are also EM derivatives that include a stochastic element quite similar to DA (see McLachlan and Krishnan, 1997).

Bound and Collapse (BC) introduced by Ramoni and Sebastiani (2001) is a two-phase algorithm. The bound phase considers possible completions of the data sample, and based on that computes an interval for each parameter estimate of the Bayesian network. The collapse phase computes a convex combination of the interval bounds, where the weights in the convex combination are computed from the available cases. The collapse phase seems to work quite well for particular missing data mechanisms but unfortunately is not guaranteed to give valid results for ignorable mechanisms in general.

Learning models from incomplete data so to speak adds a layer on top of the parameter learning methods described above. For EM, Friedman (1998) showed that doing a model selection search *within* EM will result in the best model in the limit accoring to some model scoring criterion. The Structural EM (SEM) algorithm is in essence similar to EM, but instead of computing expected sufficient statistics from the same Bayesian network model throughout the iterations, a model selection step is employed. To select the next model, a model search is performed, using the expected sufficient statistics obtained from the current model and current parameter values.

Ramoni and Sebastiani (1997) describe how BC can be used in a model selection setting. As remarked before however, BC is not guaranteed to give valid results for ignorable mechanisms in general, and the risk of obtaining invalid results unfortunately increases when the model structure is not fixed.

In contrast to SEM, our aim is not to select a single model, but to obtain a posterior probability distribution over models that correctly reflects uncertainty, including uncertainty due to missing data. Therefore our approach is more related to the simulation based DA described above.

3 Learning from complete data

In this section we discuss the Bayesian approach to learning Bayesian networks from complete data. First we introduce some notation. Capital letters denote discrete random variables, and lower case denotes a state. Boldface denote random vectors and vector states. We use $\Pr(\cdot)$ to denote probability distributions (or densities) and probabilities. $\mathcal{D} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_c)$ denotes the multinomial data sample with c i.i.d. cases. A Bayesian network (BN) for $\boldsymbol{X} = (X^1, \ldots, X^p)$ represents a joint probability distribution. It consists of a directed acyclic graph (DAG) m, called the model, where every vertex corresponds to a variable X^i, and a vector of conditional probabilities $\boldsymbol{\theta}$, called the parameter, corresponding to that model. The joint distribution factors recursively according to m as $\Pr(\boldsymbol{X}|m, \boldsymbol{\theta}) = \prod_{i=1}^{p} \Pr(X^i|\boldsymbol{\Pi}(X^i), \boldsymbol{\theta})$, where $\boldsymbol{\Pi}(X^i)$ is the parent set of X^i in m.

Since we learn BNs from a Bayesian point of view, model and parameter are treated as random variables M and $\boldsymbol{\Theta}$. We define distributions on parameter space $\Pr^{\boldsymbol{\Theta}}(\cdot)$ and model space $\Pr^{M}(\cdot)$. The superscript is omitted and we simply write $\Pr(\cdot)$ for both. The distribution on the parameter space is a product Dirichlet distribution which is conjugate for the multinomial sample $\mathcal{D}$, i.e. Bayesian updating is easy because the posterior once $\mathcal{D}$ has been taken into consideration is again Dirichlet, but with updated hyper parameters. The MAP model is found by maximizing with respect to M

$$\Pr(M|\mathcal{D}) \propto \Pr(\mathcal{D}|M) \cdot \Pr(M) \tag{1}$$

where $\Pr(\mathcal{D}|M)$ is the normalizing term in Bayes theorem when calculating the posterior Dirichlet

$$\Pr(\mathcal{D}|M) = \int \Pr(\mathcal{D}|M, \boldsymbol{\Theta}) \Pr(\boldsymbol{\Theta}|M) d\boldsymbol{\Theta} \tag{2}$$

where $\Pr(\mathcal{D}|M,\boldsymbol{\Theta})$ is the likelihood, and $\Pr(\boldsymbol{\Theta}|M)$ is the product Dirichlet prior. In Cooper and Herskovits (1992) a closed formula is derived for (2) as a function of the sufficient statistics for $\mathcal{D}$ and prior hyper parameters of the Dirichlet. This score can be written as a product of terms each of which is a function of a vertex and its parents. This decomposability allows local changes of the model to take place without having to recompute the score for the parts that stay unaltered, that is, only the score for vertices whose parents set has changes needs to be recomputed.

Instead of the MAP model, we may be interested in the expectation of some quantity Δ of models using $\Pr(M|\mathcal{D})$ as a measure of uncertainty over all models

$$\mathrm{E}[\Delta] = \sum_M \Delta_M \cdot \Pr(M|\mathcal{D}) \approx \frac{1}{q}\sum_{i=1}^{q} \Delta_{m^i} \qquad (3)$$

where the Monte Carlo approximation is obtained by sampling $\{m^i\}_{i=1}^q$ from $\Pr(M|\mathcal{D})$.

It is infeasible to calculate the normalizing factor $\Pr(\mathcal{D})$ required to obtain equality in equation (1). Madigan and York (1995) and Giudici and Castelo (2003) propose to use (enhanced) Markov chain Monte Carlo Model Composition (eMC^3) for drawing models from this distribution leaving the calculation of the normalizing term implicit. It is a sampling technique based on Markov chain Monte Carlo Metropolis-Hastings sampling summarized in the following iterated steps. At entrance assume model m^t:

1. Draw model m^{t+1} from a proposal distribution $\Pr(M|m^t)$ resulting in a slightly modified model compared to m^t (addition, reversal or removal of an arc).
2. The proposed model m^{t+1} is accepted with probability

   $$\alpha(m^{t+1}, m^t) = \min\left\{1, \frac{\Pr(\mathcal{D}|m^{t+1})\Pr(m^{t+1})}{\Pr(\mathcal{D}|m^t)\Pr(m^t)}\right\},$$

   otherwise the proposed model is rejected and $m^{t+1} \stackrel{\text{def}}{=} m^t$.

For $t \to \infty$ the models can be considered samples from the invariant distribution $\Pr(M|\mathcal{D})$. Note that in step two the normalizing factor $\Pr(\mathcal{D})$ has been eliminated. For enhanced MC^3 a third step is required, *Repeated Covered Arc Reversals* (RCAR) which simulates the neighbourhood of equivalent DAG models. We refer to Kocka and Castelo (2001) for details.

4 Learning from incomplete data

Running standard eMC^3 can be quite slow, especially for large models and data sets. In the presence of missing data, a prediction 'engine' (predicting missing components) so to speak has to be wrapped around eMC^3. Obtaining a prediction engine which will always make the correct predictions is infeasible to construct, and when the engine itself has to adapt to the ever changing model this becomes even worse. An approximate predictive engine is usually easier to construct, but will obviously sometimes make slightly wrong predictions. In this section we show how approximation can be used together with eMC^3 to obtain realizations from the posterior model distribution such that prediction errors are corrected for.

Our goal is to compute (3) when we have missing data. To be more precise, if we write $\mathcal{D} = (\mathcal{O},\mathcal{U})$ to denote the observed part $\mathcal{O}$ and the unobserved part $\mathcal{U}$, our goal is to get draws from $\Pr(M|\mathcal{O})$ such that we can use the approximation in (3). Due to incompleteness the integral in (2) no longer has a tractable solution. Our approach is instead to rewrite the posterior model distribution such that U can be "summed out" by way of "filling in". Note that the desired model posterior can be written as

$$\Pr(M|\mathcal{O}) = \sum_U \Pr(M|\mathcal{O},U)\Pr(U|\mathcal{O}). \qquad (4)$$

The first term is the distribution given in (1) involving the prior and the marginal likelihood (2). The second part is the predictive distribution which can be considered the predictor of the missing data based on the observed part. We explicitly model the predictive distribution as a BN with model M', the *imputation* model. We therefore write

$$\Pr(M|\mathcal{O},M') = \sum_U \Pr(M|\mathcal{O},U,M')\Pr(U|\mathcal{O},M') \quad (5)$$

We assume that M is independent of M' given O and U, i.e. once we are presented with complete data, the imputation model has become irrelevant

$$\Pr(M|O,U,M') = \Pr(M|O,U).$$

The Monte Carlo approximation of (5) is calculated as

$$\Pr(M|\mathcal{O},M') \approx \frac{1}{n}\sum_{i=1}^{n} \Pr(M|\mathcal{O},\mathcal{U}^i)$$

where $\mathcal{U}^i \sim \Pr(U|\mathcal{O},M')$ for $i = 1,\ldots,n$. So, if we could compute realizations from this predictive distribution we could approximate $\Pr(M|\mathcal{O},M')$. Unfortunately we can not use simple sequential Bayesian updating (Spiegelhalter and Lauritzen, 1990) for determining $\Pr(U|\mathcal{O},M')$. Instead of sampling from the true predictive distribution, we define an *approximate predictive distribution* which can act as a proposal distribution for suggesting imputations. The predictive

distribution can be rewritten such that it can act as a quality measure for a proposed imputation. This is accomplished by using *importance sampling*. Denote the approximate predictive distribution $\Pr^*(U)$ and rewrite (5)

$$\Pr(M|\mathcal{O},M') = \sum_U \Pr(M|\mathcal{O},U)\frac{\Pr(U|\mathcal{O},M')}{\Pr^*(U)}\Pr^*(U)$$

Sample $\mathcal{U}^i \sim \Pr^*(U)$ for $i = 1,\ldots,n$ and use that sample in the importance sampling approximation

$$\Pr(M|\mathcal{O},M') \approx \frac{1}{W}\sum_{i=1}^{n} \underbrace{\frac{\Pr(\mathcal{U}^i|\mathcal{O},M')}{\Pr^*(\mathcal{U}^i)}}_{w_i} \Pr(M|\mathcal{O},\mathcal{U}^i) \quad (6)$$

where $W = \sum_{i=1}^{n} w_i$ is the normalizing constant. Now rewrite the predictive distribution

$$\Pr(\mathcal{U}^i|\mathcal{O},M') = \Pr(\mathcal{U}^i,\mathcal{O}|M')\frac{1}{\Pr(\mathcal{O}|M')}$$

where the term $\Pr(\mathcal{U}^i,\mathcal{O}|M')$ is given by (2). Through normalization the denominator $\Pr(\mathcal{O}|M')$ disappears as it is independent of U. It therefore suffices to calculate the weights as

$$w_i = \frac{\Pr(\mathcal{U}^i,\mathcal{O}|M')}{\Pr^*(\mathcal{U}^i)} \quad (7)$$

where the numerator can be computed efficiently and the denominator is the probability of the proposal. The marginal likelihood in the numerator is in this context not used as a scoring criterion for models. Instead we use it as a scoring criterion for *imputations*. The denominator compensates for the bias introduced by drawing from $\Pr^*(U)$ rather than the correct predictive distribution.

Given a set $\{\mathcal{U}^i\}_{i=1}^n$ that has been sampled from $\Pr^*(U)$, sampling from the mixture approximation (6) of $\Pr(M|\mathcal{O},M')$ is done as follows:

1. The probability of selecting sample $\mathcal{U}^i$ augmenting $\mathcal{O}$ is proportional to the importance weight w_i.
2. The now complete sample $(\mathcal{O},\mathcal{U}^i)$ is used for sampling models from $\Pr(M|\mathcal{O},\mathcal{U}^i)$ using eMC^3.

We can now draw from $\Pr(M|\mathcal{O},M')$, but our goal was to obtain draws from $\Pr(M|\mathcal{O})$, i.e. we need samples from the posterior model distribution given observed data *without* conditioning on the imputation model M'. The desired distribution is obtained by Gibbs sampling. Given an imputation model m^l draw the following

$$\begin{aligned} m^{l+1} &\sim \Pr(M|\mathcal{O},m^l) \\ &\vdots \end{aligned}$$

Algorithm e**MC**$^4(n,q,k)$

```
1  m^0 ← G = (V = X, E = ∅)
2  r ← 0
3  for l ← 0 to k
4      W ← 0
5      U^0 ← U^r
6      for i ← 0 to n
7          w_i ← Pr(O,U^i|m^l) / Pr*(U^i)
8          W ← W + w_i
9          if i ≠ n then draw U^{i+1} ~ Pr*(U)
10     draw r ~ Pr(i) = w_i/W
11     m^0 ← m^l
12     for t ← 0 to q
13         m^t ← RCAR(m^t)
14         draw m^{t+1} ~ Pr(M|m^t)
15         B ← Pr(O,U^r|m^{t+1}) / Pr(O,U^r|m^t)
16         draw α ~ Bernoulli(min{1, B})
17         if α ≠ 1 then m^{t+1} ← m^t
18     m^{l+1} ← m^{q+1}
19     LogToFile(m^{l+1})
```

Figure 1: The eMC^4 algorithm

which for $l \to \infty$ results in a chain of realizations from $\Pr(M|\mathcal{O})$. This in effect allows us to calculate (3).

When n is large, the mixture approximation is close to the real distribution $\Pr(M|\mathcal{O},M')$. However, the invariant model distribution is reached for any value assigned to n if we make sure that one of the imputation proposals is the *current* imputation, i.e. the augmented data sample that was selected at the last mixture draw before entering the eMC^3 loop. By this overlap, n is indirectly increased every time the mixture is set up. From a practical point of view however, n does have an impact on how well the model Markov chain mixes. Small n implies slow mixing depending on how far the approximate predictive distribution is from the real predictive distribution.

We do not discuss parameter estimation in this paper, but merely mention that using the importance sampler presented above, it is also possible to approximate the posterior *parameter* distribution. In (6) simply plug in $\Pr(\boldsymbol{\Theta}|M',\mathcal{O},\mathcal{U}^i)$ instead of $\Pr(M|\mathcal{O},\mathcal{U}^i)$ to obtain the required posterior.

To summarize, figure 1 contains the pseudo-code for the algorithm called *enhanced Markov Chain Monte Carlo Model Composition with Missing Components*, or for short, eMC^4. In line 1 an initial empty graph is defined. In lines 6–9 the imputations take place and the importance weights are calculated. Line 10 is the first step in drawing from the mixture. Lines

12–17 perform the eMC^3 algorithm based on the augmented sample selected. In the absence of relevant prior knowledge, a uniform model prior is assumed in line 15. Angelopoulos and Cussens (2001) discuss the construction of informative model priors. The choice of k (number of iterations of the Gibbs model sampler) depends on when the Markov chain of models has converged. Monitoring the average number of edges is one method for doing so suggested by Giudici and Green (1999). Once this average stabilizes the chain has converged.

5 Proposal distribution $\Pr^*(U)$

We can choose freely the approximate predictive distribution from which samples are drawn for (6), but of course some choices are better than others. Ideally, the appoximate predictive distribution should be close to the real predictive distribution, because otherwise n is required to be large to obtain enough samples from the region where the mass of the real distribution is located. Existing imputation techniques can be used, as long as they can be cast in the form of a distribution from which imputations can be drawn. Naturally it is also a requirement that $\Pr^*(U)$ has support when $\Pr(U|\mathcal{O}, M')$ has support. A uniform proposal distribution is probably unwise unless a very small fraction of data is missing. On the other hand, a distribution based on M' with parameters estimated using EM is not needed. Employing the BC algorithm for parameter estimation using M' could be interesting, since it is fast and is reported to often give reasonable results. Alternatively, a simple *available cases analysis* may prove to be good enough.

It is not a requirement to use the actual imputation model M' as the basis for $\Pr^*(U)$. In fact $\Pr^*(U)$ need not be modeled as a BN at all. However, an imputation method that does not take the independences portrayed by M' into account will have a hard time proposing qualified imputations, because the degree of freedom is simply too high (assuming that nothing is known about the missing data mechanism). Similarly, the parameter does not need to be derived from the data; however, since the missing data mechanism is assumed to be ignorable, all information we need to impute (predict) is contained (indirectly) in $\mathcal{O}$, and therefore predictions should at least depend on observed values. We propose to model $\Pr^*(U) \stackrel{\text{def}}{=} \Pr(U|\mathcal{O}, M', \boldsymbol{\theta})$ as a BN with model M' and parameter $\boldsymbol{\theta} = \mathrm{E}[\boldsymbol{\Theta}|\mathcal{O}, \mathcal{U}^{M'}]$, where expectation is with respect to $\Pr(\boldsymbol{\Theta}|M')$, and $(\mathcal{O}, \mathcal{U}^{M'})$ is the augmented sample from which M' was learned. The latter makes sense because $(\mathcal{O}, \mathcal{U}^{M'})$ is the sample from which the model was learned and therefore reflects the most appropriate sample on which to base the parameter.

In order do draw multivariates we propose the following method based on Gibbs sampling, where realizations are drawn on a univariate level. Denote a case j in $\mathcal{D}$ by $\boldsymbol{d}_j = (x_j^1, \ldots, x_j^p) = (\boldsymbol{o}_j, \boldsymbol{u}_j) = (\boldsymbol{o}_j, (u_j^1, \ldots, u_j^{r(j)}))$, where $\boldsymbol{o}_j$ and $\boldsymbol{u}_j$ refer to the observed and unobserved part of the case. The j'th case for $\mathcal{U}^t$ is sampled as follows

$$\begin{aligned} u_j^{1,t} &\sim \Pr(U_j^1 | u_j^{2,t-1}, \ldots, u_j^{r(j),t-1}, \boldsymbol{o}_j, M', \boldsymbol{\theta}) \\ &\vdots \\ u_j^{r(j),t} &\sim \Pr(U_j^{r(j)} | u_j^{1,t}, \ldots, u_j^{r(j)-1,t}, \boldsymbol{o}_j, M', \boldsymbol{\theta}). \end{aligned}$$

Based on Markov chain Monte Carlo theory, correlated multivariate realizations $\boldsymbol{u}_j \sim \Pr(\boldsymbol{U}_j|\boldsymbol{o}_j, M', \boldsymbol{\theta})$ are obtained when $t \to \infty$. Since each draw in the Gibbs sampler is univariate, and the entire Markov blanket of variable U^i has evidence, inference does not require any advanced techniques.

With the suggested Gibbs sampler we effectively collect *all realizations* including samples in the burn-in phase. The idea is to let the importance sampler decide on the quality of the proposed imputations.

We can calculate the importance weights efficiently without explicitely knowing the actual probabilities $\Pr(\mathcal{U}^i|\mathcal{O}, M', \boldsymbol{\theta})$. This can be seen by rewriting the importance weights from (7):

$$\begin{aligned} w_i &= \frac{\Pr(\mathcal{U}^i, \mathcal{O}|M')}{\Pr(\mathcal{U}^i|\mathcal{O}, M', \boldsymbol{\theta})} \\ &= \frac{\Pr(\mathcal{U}^i, \mathcal{O}|M') \cdot \Pr(\mathcal{O}|M', \boldsymbol{\theta})}{\Pr(\mathcal{U}^i, \mathcal{O}|M', \boldsymbol{\theta})}. \end{aligned}$$

By normalization of these weights, $\Pr(\mathcal{O}|M', \boldsymbol{\theta})$ cancels out, and it suffices to calculate the weights as

$$w_i = \frac{\Pr(\mathcal{U}^i, \mathcal{O}|M')}{\Pr(\mathcal{U}^i, \mathcal{O}|M', \boldsymbol{\theta})},$$

the ratio of the marginal likelihood over the likelihood given $\boldsymbol{\theta}$. This ratio is easily obtained because both probabilites can be computed efficiently in closed form. In summary, we can propose imputations efficiently *and* we can compute the "quality" of such a propsal efficiently as well.

When the structural difference between the imputation model M' and the exit model M is kept relatively small (dependent on q), we can make the following observations when $\boldsymbol{\theta}$ is assigned $\mathrm{E}[\boldsymbol{\Theta}|\mathcal{O}, \mathcal{U}^{M'}]$:

(1) Multivariate *predictions* based on the two models are correlated. Hence n need not be large in order to compensate for the predictive difference. However, we may still need many realizations $\mathcal{U}^i$ in order to

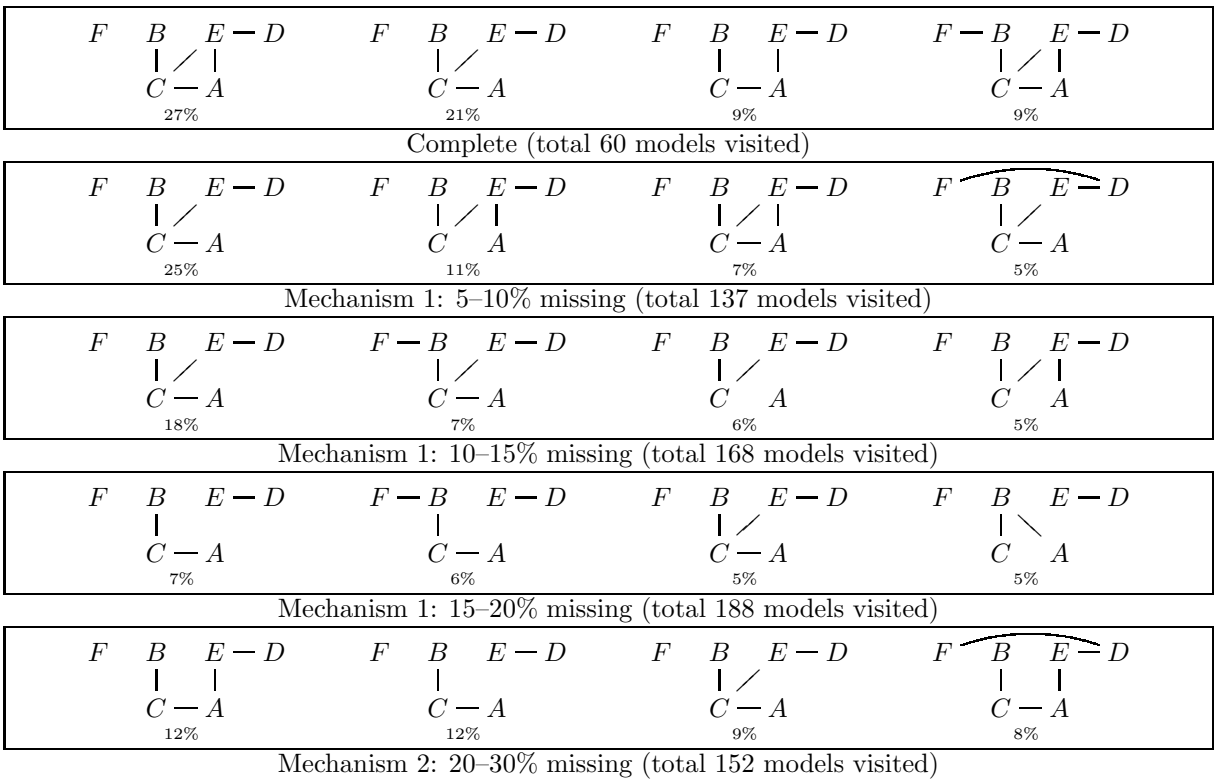


Figure 2: Top four visited models. Note that all edges are reversible.

capture the entire distribution. Correlation between multivariate predictions allows us to keep n relatively small and only move slightly in a direction towards a 'better' prediction.

(2) Because models are correlated and as a consequence also the predictions, the first predictions obtained by running the Markov blanket Gibbs sampler may be good. The Gibbs sampler so to speak picks up from where it left the last time and continues imputation using the new model. This means that the there is a fair chance that an initial imputation is actually selected.

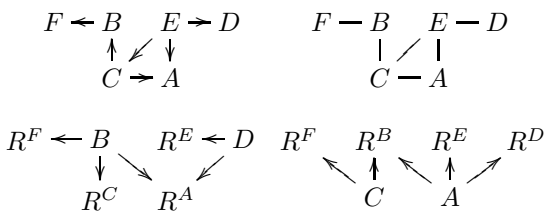


Figure 3: *Top*: Generating model (left), Essential graph (right). *Bottom*: Mechanism 1 (left), Mechanism 2 (right).

6 Experimental evaluation

In this section we perform a small experimental evaluation of eMC^4 and briefly discuss the results. Because our approach is Bayesian, comparison of the results with model *selection* methods for incomplete data such as SEM is not very useful.

We used a data set from Edwards and Havránek (1985) about probable risk factors of coronary heart disease. The data set consists of 1841 records and 6 binary variables, A: smoking, B: strenuous mental work, C: strenuous physical work, D: blood pressure under 140, E: ratio β to α proteins less than 3, F: family history of coronary heart disease. Because several DAGs encode the same set of assumptions about independence, we depict results as *essential graphs*, a canonical representation of an equivalence class (see Chickering, 1995; Kocka and Castelo, 2001).

Based on an eMC^3 run using the 1841 complete records, the 1st model in figure 3 is a highly probable model (although edge $F-B$ is not strongly supported). The 2nd model in the figure is the corresponding essential graph of the DAG. The parameter corresponding to the DAG model was determined based on the aforementioned data set, and 1800 new records were sampled from the BN. Incomplete sets were generated by

applying missingness mechanism one in figure 3 on the complete sample. This graph explicitly defines how response R^i of variable i depends on observed variables. Since for all i, R^i only depends on completely observed variables, the missingness mechanism is clearly ignorable. Three incomplete sets were generated with 5–10%, 10–15% and 15–20% missing components. The probability of non-response of variable i conditional on a parent configuration of R^i was selected from the specified interval.

On the basis of the generating model and the missingness mechanism, we would expect the following results. Since response of C only depends on B and the association $C - B$ is strong, a big fraction of components can be deleted for C without destroying support for the edge in the data. Association $D - E$ is also strong so discarding components for E will probably not have a major impact on the edge either. Association $E - A$ is influenced by B and D because the response is determined by those variables. Values for E and A may be absent often and therefore information about the association might have changed. This may also be the case for the edges $C - E$ and $C - A$.

We ran eMC^4 using each incomplete data set. Parameter q (number of eMC^3 iterations) was set to 150 and n to 25 (number of imputations). It took about 15 minutes on a 2 GHz machine before the Markov chain appeared to have converged.

In figure 2 the top four models are depicted along with their sampling frequencies. Notice the presence of the strong associations $C - B$ and $D - E$ everywhere, as expected. When the fraction of missing components for two associated variables increases it has a big impact on the support of such an association. Indeed, from the figure we see that the support for associations between variables A, C and E has changed. The sample frequencies and the number of visited models also suggest that the variance of the posterior distribution becomes bigger when more components are deleted. There is no longer a pronounced 'best' model.

The plot in figure 4 shows this more clearly. Here the cumulative frequencies are plotted against models (sorted on frequency in descending order). A steep plot indicates a small variance. For complete data the 10 best models account for 90% of the distribution whereas for 15–20% missing components only 50% of the distribution is accounted for by the best 10 models. To investigate the similarity of the models between the three incomplete sets, we used equation (3) to calculate Δ, where Δ_M is set to 1 when there is an edge between two vertices of interest in M. This results in the expected probability of the presence of edges as seen in figure 4. We can see, as we would expect,

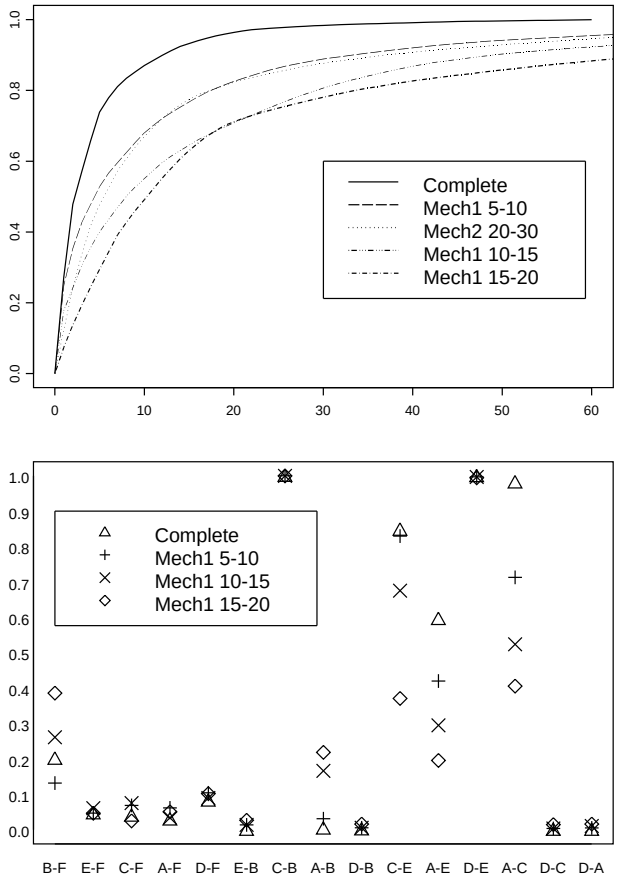


Figure 4: *Top*: Cumulative frequencies. *Bottom*: Expected probability of edges.

that the distance between points of complete data and incomplete data is dependent on the fraction of missing components. Diamonds (15–20%) have the biggest distance to triangles (complete), and plusses (5–10%) the smallest.

As we saw for mechanism one, discarding components for two associated variables can have a big impact on the presence of the corresponding edge in sampled models. For strongly associated variables the impact is less pronounced. We created another incomplete data set using mechanism two in figure 3. For the associated variables C, E and A the mechanism only discards components that we think will not severely impact these associations. For the strong association $E - D$ discarding components on both should not matter. We expect that we are able to remove a substantial fraction of components and still obtain reasonable models. We selected the fraction of missing components in the interval 20–30%. In the last row in figure 2 we see that although a substantial fraction of components were deleted, the models learned are quite similar to the models from the complete set.

To illustrate that it is not the fraction of missing components that determines the variance but rather the fraction of missing information (Little and Rubin, 1987), we plotted the cumulative frequency in figure 4. The variance of the posterior distribution is similar to the variance of the posterior for mechanism one with 5–10% missing components. This means that although the fraction of missing components is much higher than 5–10%, the uncertainty due to missing data has not changed substantially.

7 Conclusion

We have presented eMC^4 for simulating draws from the posterior distribution of BN models given incomplete data. In contrast to existing methods for BN model learning with incomplete data, we take a Bayesian approach and approximate the posterior model distribution given the observed data. Different imputation methods may be used, and specifically we describe a method that does not require exact inference in a BN. By using importance sampling we give all multivariate realizations of the Markov chain a 'chance' of being selected rather than just returning the last realization as in traditional Gibbs sampling. Importance sampling makes it possible to exploit qualified, yet not perfect imputation proposals. From a computational point of view specifying an approximate distribution is cheaper than a perfect one.

Valuable insight is gained when sampling models from the posterior; an illustration of the kind of information one can derive from posterior realizations is given in section 6. A posterior distribution is more informative than just a single model. This is especially true in the case of incomplete data, since the increased uncertainty due to missing data is reflected in the probability distribution.

References

N. Angelopoulos and J. Cussens. Markov chain Monte Carlo using trees-based priors on model structure. In J. Breese and D. Koller, editors, *Proc. of the Conf. on Uncertainty in AI*, pages 16–23, 2001.

D. Chickering. A transformational characterization of equivalent Bayesian networks. In P. Besnard and S. Hanks, editors, *Proc. of the Conf. on Uncertainty in AI*, pages 87–98, 1995.

G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Statistical Society, Series B*, 34:1–38, 1977.

D. Edwards and T. Havránek. A fast procedure for model search in multidimensional contingency tables. *Biometrika*, 72(2):339–351, 1985.

N. Friedman. The Bayesian structural EM algorithm. In G. F. Cooper and S. Moral, editors, *Proc. of the Conf. on Uncertainty in AI*, pages 129–138, 1998.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(6), 1984.

P. Giudici and R. Castelo. Improving Markov chain Monte Carlo model search for data mining. *Machine Learning*, 50(1):127–158, 2003.

P. Giudici and P. Green. Decomposable graphical gaussian model determination. *Biometrika*, 86(4): 785–801, 1999.

T. Kocka and R. Castelo. Improved learning of Bayesian networks. In D. Koller and J. Breese, editors, *Proc. of the Conf. on Uncertainty in AI*, pages 269–276, 2001.

S. L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201, 1995.

R. J. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. Wiley and Sons, 1987.

D. Madigan and J. York. Bayesian graphical models for discrete data. *Intl. Statistical Review*, 63:215–232, 1995.

G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, 1997.

M. Ramoni and P. Sebastiani. Learning Bayesian networks from incomplete databases. In D. Geiger and P. Shenoy, editors, *Proc. of the Conf. on Uncertainty in AI*, pages 401–408, 1997.

M. Ramoni and P. Sebastiani. Robust learning with missing data. *Machine Learning*, 45(2):147–170, 2001.

D. J. Spiegelhalter and S. L. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20:579–605, 1990.

M. Tanner and W. Wong. The calculation of posterior distributions by data augmentation. *J. of the Am. Stat. Assoc.*, 82(398):528–540, 1987.

On the Behavior of MDL Denoising

Teemu Roos **Petri Myllymäki**
Helsinki Institute for Information Technology
Univ. of Helsinki & Helsinki Univ. of Technology
P.O. Box 9800 FIN-02015 TKK, Finland

Henry Tirri
Nokia Research Center
P.O. Box 407 Nokia Group
FIN-00045, Finland

Abstract

We consider wavelet denoising based on minimum description length (MDL) principle. The derivation of an MDL denoising criterion proposed by Rissanen involves a renormalization whose effect on the resulting method has not been well understood so far. By inspecting the behavior of the method we obtain a characterization of its domain of applicability: good performance in the low variance regime but over-fitting in the high variance regime. We also describe unexpected behavior in the theoretical situation where the observed signal is pure noise. An interpretation for the renormalization is given which explains both the empirical and theoretical findings. For practitioners we point out two technical pitfalls and ways to avoid them. Further, we give guidelines for constructing improved MDL denoising methods.

1 INTRODUCTION

Most natural signals such as audio and images are typically redundant in that the neighboring time-slots or pixels are highly correlated. Wavelet representations of such signals are very sparse, meaning that most of the wavelet coefficients are very small and the information content is concentrated on only a small fraction of the coefficients (Mallat, 1989). This can be exploited in data compression, pattern recognition, and denoising, i.e., separating the informative part of a signal from noise. In statistics the denoising problem has been analyzed in terms of statistical risk, i.e., the expected distortion under an assumed model where typically distortion is defined as squared error and the model consists of deterministic signal plus additive Gaussian noise. Donoho & Johnstone (1994) prove that certain thresholding methods are nearly minimax optimal for a large class of signals. In the Bayesian approach a prior distribution is postulated for the signal and the expected (Bayes) risk is minimized (Ruggeri & Vidakovic, 1999). Both approaches require that parameters such as noise variance are known beforehand or determined as a part of the process.

The minimum description length (MDL) philosophy offers an alternative view where the noise is *defined* as the incompressible part of the signal (Rissanen, 2000). We analyze Rissanen's MDL denoising method and characterize its domain of applicability. We show that the method performs well in the low variance regime but fails in the high variance regime when compared to a thresholding method proposed by Donoho and Johnstone. In particular, in the theoretical situation where the noise completely dominates the signal, the MDL denoising method retains a majority of the wavelet coefficients even though in this case discarding all coefficients is the optimal solution in terms of both statistical risk and what we intuitively understand as separating information from noise.

We explain the behavior of the MDL method by showing that it results not from the MDL principle itself but from a renormalization technique used in deriving the method. We also point out two technical pitfalls in the implementation of MDL denoising that practitioners should keep in mind. Further, we give guidelines for constructing MDL denoising methods that have a wider domain of applicability than the current one and list objectives for future research in this direction.

2 MDL PRINCIPLE

We start by introducing some notation and briefly reviewing some of the relevant parts of MDL theory. A recent introduction to MDL is given by Grünwald (2005), see also Barron *et al.* (1998).

2.1 STOCHASTIC COMPLEXITY

Let y^n be a sequence of observations. We define a model class as a set of densities $\{f(y^n\,;\theta)\colon \theta\}$ indexed by a finite-dimensional parameter vector θ. The maximum likelihood estimator of the parameter vector is denoted by $\hat{\theta}(y^n)$. The normalized maximum likelihood (NML) density for a model class parameterized by parameter vector θ is defined by

$$\bar{f}(y^n) = \frac{f(y^n\,;\hat{\theta}(y^n))}{C^n}, \tag{1}$$

where C^n is a normalizing constant:

$$C^n = \int_Y f(y^n\,;\hat{\theta}(y^n))\; dy^n. \tag{2}$$

Implicit in the notation is the range of integration Y within which the data y^n is restricted. A range other than the full domain of y^n is necessary in cases where the integral is otherwise unbounded.

The difference between the ideal code-length (negative logarithm) of the NML density and the unachievable maximum likelihood code-length is given by the *regret* which is easily seen to be constant for all data sequences y^n:

$$-\ln \bar{f}(y^n) - [-\ln f(y^n\,;\hat{\theta}(y^n))] = \ln C^n.$$

The NML density is the unique minimizer in Shtarkov's minimax problem (Shtarkov, 1987):

$$\min_q \max_{y^n} -\ln q(y^n) - [-\ln f(y^n\,;\hat{\theta}(y^n))] = \ln C^n,$$

and the following more general problem:

$$\min_q \max_p E_p - \ln q(y^n) - [-\ln f(y^n\,;\hat{\theta}(y^n))] = \ln C^n,$$

where the expectation over y^n is taken with respect to the worst-case data generating density p. For any density q other than the NML density, the maximum (expected) regret is greater than $\ln C^n$. Further, the NML is also the *least favorable* distribution in that is the unique maximizer of the maximin problem with the order of the min and max operators in the latter problem above exchanged. For these reasons the NML code is said to be universal in that it gives the shortest description of the data achievable with a given model class, deserving to be defined as the *stochastic complexity* of the data for the model class. The MDL principle advocates the choice of the model class for which stochastic complexity is minimized.

2.2 PARAMETRIC COMPLEXITY

It is instructive to view NML as seeking a balance between fit versus complexity. The numerator measures how well the best model in the model class can represent the observed data while the denominator 'penalizes' too complex model classes. The logarithm of the denominator, $\ln C^n$, is termed *parametric complexity* of the model class. Currently one of the most active areas of research within the MDL framework is the problem of unbounded parametric complexity which makes it impossible to define the NML density for models such as geometric, Poisson, and Gaussian families, see (Grünwald, 2005).

For model classes with unbounded parametric complexity, Rissanen (1996) proposes to use a two-part scheme where the range of the data is first encoded using a code based on an universal code for integers after which the data is encoded using NML taking advantage of the restricted range. Foster & Stine (2001, 2005) analyze similar schemes where the range of the *parameters* is restricted instead that of the the data. A weakness in such solutions is that they typically result in two-part codes that are not complete, i.e., the corresponding density integrates to less than one.

Rissanen (2000) describes an elegant renormalization scheme where the hyperparameters defining the range of the data are optimized and a second normalization is performed such that the resulting code is complete. This 'renormalized' NML can be used for model selection in linear regression and denoising. We discuss the renormalization and the resulting MDL denoising criterion more thoroughly in Sec. 4.

3 WAVELET DENOISING

Wavelet denoising can be seen as a special case of linear regression with regressor selection. For a good textbook on wavelets, see (Daubechies, 1992). An ex-

tensive review of statistical uses of wavelets is given by Abramovich *et al.* (2000).

3.1 WAVELET REGRESSION

This section closely follows Rissanen (2000). Let X be an $n \times k$ matrix of *regressor variables* (independent variables), and y^n be a vector of n *regression variables* (dependent variables). In a linear regression model the regression variables are dependent on the regressor variables and a $k \times 1$ parameter vector β through the equation $y^n = X\beta + \epsilon^n$, where ϵ^n is a vector of n noise terms that are modeled as independent Gaussian with zero mean and variance σ^2. This is equivalent to the equation

$$f(y^n; \beta, \sigma) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{\|y^n - X\beta\|^2}{2\sigma^2}\right), \quad (3)$$

where $\|\cdot\|^2$ denotes the squared Euclidean norm. The regressor matrix X is considered fixed and given in all of the following and therefore omitted in the notation. We define the matrices $Z = X'X$ and $\Sigma = n^{-1}Z$ which are assumed to be positive definite in order to guarantee uniqueness of maximum likelihood estimates. The maximum likelihood estimators of β and σ^2 are independent and given by

$$\hat{\beta}(y^n) = Z^{-1}X'y^n, \quad (4)$$

$$\hat{\sigma}^2(y^n) = \frac{1}{n}\|y^n - X\hat{\beta}'(y^n)\|^2. \quad (5)$$

Now, assume the vector y^n can be considered a series, i.e., the data points are ordered in a meaningful way. We can then obtain a regressor matrix X by various transformations of the index i of the y_i variables. Thus, we define for each $j \leq k$, $X_{i,j} = f_j(i)$, where f_j are arbitrary basis functions. One both theoretically and practically appealing way to define the functions f_j is to use a *wavelet basis*, see e.g., Daubechies (1992). By letting the regressor matrix be square, i.e., $k = n$, and taking as the basis functions $f_j(i)$ an appropriate wavelet basis, we get an *orthogonal* regressor matrix X, i.e., X has as its inverse the transpose X' and we have $Z = X^{-1}X = I$, where I is the identity matrix.

Instead of using all the basis vectors, we may also choose a subset γ of them. This gives the reconstructed version $\hat{y}^n_\gamma = X\hat{\beta}_\gamma(y^n)$, and the difference to the original signal is left to be modeled as noise. Since the basis is orthogonal, the maximum likelihood values of any subset of all the parameters are equal to the corresponding maximum likelihood parameters in the full model and one gets the parameter vector

$$\hat{\beta}_\gamma(y^n) = (\delta_i(\gamma)\hat{\beta}_i(y^n))',$$

where $\delta_i(\gamma)$ is equal to one if the index i is in the index set γ of retained coefficients and zero otherwise. The maximum likelihood estimator of the noise variance becomes

$$\begin{aligned}\hat{\sigma}^2_\gamma(y^n) &= \frac{1}{n}\|X\hat{\beta}'(y^n) - X\hat{\beta}'_\gamma(y^n)\|^2 \\ &= \frac{1}{n}\|\hat{\beta}(y^n) - \hat{\beta}_\gamma(y^n)\|^2,\end{aligned}$$

which is seen to be the sum of the discarded coefficients divided by n. We denote for convenience the squared norm of the maximum likelihood coefficient vector corresponding to γ by S_γ:

$$S_\gamma = \|\hat{\beta}_\gamma(y^n)\|^2 = \sum_{i\in\gamma}\beta_i^2.$$

The squared norm of the coefficient in the full model with $k = n$ is denoted simply by S. From orthogonality it follows that S is equal to the squared norm of the data $\|y^n\|^2$.

3.2 THE DENOISING PROBLEM

The denoising problem is now to choose a subset γ such that the retained coefficients would give a good reconstruction of the informative part of the signal while the discarded coefficients would contain as much of the noise in the signal as possible, The sparseness of wavelet representations, i.e., the fact that a large fraction of the coefficients are essentially zero in the 'noise-free' or informative part of the signal (see (Mallat, 1989)) makes it plausible to recover the informative part by identifying and discarding the coefficients that are likely to contain pure noise.

The idea of *wavelet thresholding* was proposed soon after Mallat's paper independently by Donoho & Johnstone (1991) and Weaver *et al.* (1991). In wavelet thresholding a threshold value is first determined and the coefficients whose absolute value is less than the threshold are discarded. Using the maximum likelihood estimates as the values of the retained coefficients is called *hard thresholding* while in *soft thresholding* the retained coefficients are also shrunk towards zero in order to reduce the noise distorting the informative coefficients.

In statistical wavelet denoising the denoising problem is often formalized using the concept of *statistical risk*, i.e., the expected distortion (usually squared error) of the reconstructed signal when compared to a true signal. This requires an assumed model typically involving i.i.d. noise added to a true signal. In the statistical approach the signal is considered deterministic and the worst-case risk over a class of signals is minimized while in the Bayesian approach (see, e.g., (Ruggeri & Vidakovic, 1999; Chang *et al.* , 2000)) a prior distribution on the true signal is postulated and the expected (Bayes) risk is minimized. Donoho & Johnstone (1994) have derived a set of wavelet denoising methods including the following hard threshold:

$$t_{DJ} = \sigma\sqrt{2\log n}, \tag{6}$$

where σ is the standard deviation of noise.

In order to apply the method in practice, one usually needs to estimate σ. Donoho & Johnstone suggest using as an estimator the median of the coefficients on the finest level divided by .6745 which usually works well as long as the signal is contained mainly in the low frequency coefficients. There are also several other, more refined denoising methods suggested by the mentioned authors and others but due to space limitations and the fact that our real focus is in understanding the behavior of MDL based denoising, these methods are not discussed in the current paper. Fodor & Kamath (2003) present an empirical comparison of different wavelet denoising methods; see also Ojanen *et al.* (2004) for a comparison of the Donoho-Johnstone method and MDL denoising.

4 MDL DENOISING

The MDL principle offers a different approach to denoising where the objective is to separate information and noise in the observed signal. Unlike in the statistical approach, information and noise are *defined* as the compressible and the incompressible part of the signal respectively, thus depending on the model class used for describing the signal.

4.1 MDL APPROACH TO DENOISING

One of the most characteristic features of the MDL approach to statistical modeling is that there is no need to assume a hypothetical generating model whose existence would be very hard to verify. Any background information regarding the phenomenon under study is incorporated in the choice of the model class. The only assumption is that at least one of the model classes under consideration allows compression of the data which is clearly much easier to accept than the assumption that the assumed model is indeed an exact replica of the true generating mechanism.

In denoising, MDL model selection is performed by considering each subset of the coefficients as a model class and minimizing the stochastic complexity of the data given the model class. Unfortunately, for wavelet based models and more generally, for linear regression models, the normalizer in the NML density is unbounded and NML is not defined unless the range of the data is restricted. The problem can be solved by resorting to universal models other than NML, such as two-part or mixture models in defining the stochastic complexity. Hansen & Yu (2000) propose a combination of two-part and mixture codes for wavelet denoising. Their method also includes an estimation step similar to the one used by Donoho & Johnstone, and is thus not completely faithful to the MDL philosophy.

4.2 RENORMALIZED NML

Rissanen (2000) solves the problem of unbounded parametric complexity by two-fold normalization. The data range is first restricted such that the squared (Euclidean) norm of the maximum likelihood values of the wavelet coefficients $\|\hat{\beta}_\gamma(y^n)\|^2$ is always less than some maximal value R and the maximum likelihood variance $\hat{\sigma}^2_\gamma(y^n)$ is greater than some minimal value σ_0^2. We then obtain an NML density with limited support for each pair (R, σ_0^2). It is now possible to construct a 'renormalized' or 'meta' NML density by taking the obtained NML densities as a new model class[1].

After the application of Stirling's approximation to gamma functions and ignoring constant terms it can be shown that the code-length to be minimized becomes[2]

$$\frac{(n-k)}{2}\ln\frac{S-S_\gamma}{n-k} + \frac{k}{2}\ln\frac{S_\gamma}{k} + \frac{1}{2}\ln(k(n-k)). \tag{7}$$

[1] In fact even the renormalization requires the data range to be restricted but it turns out that the final range doesn't affect the resulting criterion.

[2] Multiplying the code-length formula by two gives an equivalent minimization problem. Note the last term that was incorrect in some of the earlier publications.

It can be shown that the criterion is always maximized by choosing γ such that either the k largest or the k smallest coefficients are retained for some k. We consider this an artefact of the renormalization performed and assume in the what follows that the k largest coefficients are retained. We return to the issue in Sec. 5.3.

4.3 PRACTICAL ISSUES

We point out two issues of a rather technical nature that nevertheless deserve to be noted by practitioners since we have found them to result in very poor performance in more than one case. First, in all wavelet thresholding methods, it should be made sure that the wavelet transform used is such that the coefficients are scaled properly, in other words, that the corresponding basis is orthogonal. This is essential for all wavelet thresholding methods. It is easy to check that the sum of squares of the original data and the transformed coefficients are always equal.

Secondly, since the criterion is derived for continuous data and involves densities, problems may occur when it is applied to low-precision or discrete, say integer, data. If the data can be represented exactly by some number k_0 of coefficients, the criterion becomes minus infinity for all $k \geq k_0$ because the first term includes a logarithm of zero. Also, for k almost as large as k_0 the criterion takes a very small value and such a value of k is often selected as the optimal one potentially resulting in severe over-fitting. This problem may either be solved by using a lower bound for $(S - S_\gamma)/(n - k)$ corresponding to a lower bound on the variance. Alternatively, once a sudden drop to minus infinity in the criterion is recognized it is possible to reject all values of k that are near the point where the drop occurs.

5 BEHAVIOR OF MDL DENOISING

By inspecting the behavior of the MDL denoising criterion as a function of noise variance, we are able to give a rough characterization of its domain of applicability. This makes way towards a more important goal, the understanding of renormalized NML, and potential ways of generalizing and improving it.

5.1 EMPIRICAL OBSERVATIONS

Fig. 1 illustrates the behavior of the MDL denoising method and the method by Donoho & Johnstone described in Sec. 3 with Daubechies N=4 wavelet basis. The original image is distorted by Gaussian noise to get a noisy signal. When there is little noise, the difference is small, MDL method performing better in terms of standard error. However, when there is much noise the methods produce very different results. The Donoho-Johnstone method retains only 0.3 percent of the coefficients while the MDL method retains 46.9 percent of them, the former giving a better result in terms of standard error.

Figure 1: Lena Denoised. *Top left:* Noisy image ($\sigma = 10.0$); *middle left:* Donoho-Johnstone (2.2 % retained, std. error 8.1); *bottom left:* MDL (7.6 % retained, std. error 6.8); *top right:* Noisy image ($\sigma = 47.5$); *middle right:* Donoho-Johnstone (0.3 % retained, std. error 17.3); *bottom right:* MDL (46.9 % retained, std.error 44.9).

The effect of the standard deviation of noise on the behavior of the two methods can be clearly seen in Fig. 2. It can be seen that the MDL method outperforms the Donoho-Johnstone method when the noise standard deviation is less than 15. However, outside this range the performance of the MDL method degrades linearly

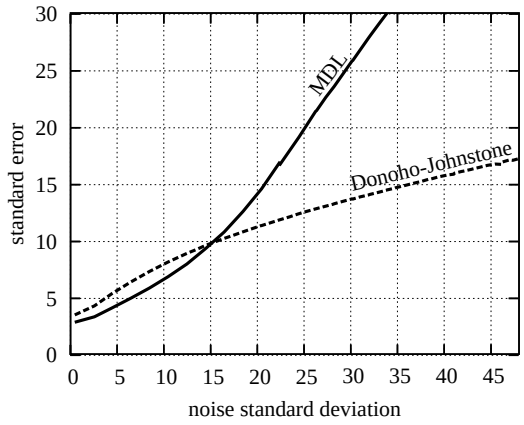


Figure 2: Effect of noise.

due to retaining too many coefficients. The standard error of the noise should be compared to the standard deviation of the original signal which in this case was 46.6. Experiments with other natural images indicate that the standard deviation of the signal determines the scale but does not affect the shape of the curves. As a rough characterization of the domain of applicability of the MDL method it can be said that the noise standard deviation should be at most half of the standard deviation of the signal.

5.2 THEORETICAL ANALYSIS

The degradation of performance of the MDL denoising criterion is underlined when the noise variance is very large. This can be demonstrated theoretically by considering what happens when the noise variance grows without bound so that in the limit the signal is pure Gaussian noise. Since the criterion is scale invariant we may without loss of generality assume unit variance. Essentially, we need to evaluate the asymptotics of S_k, the squared sum of the k largest coefficients in absolute value. Let $\beta_{i_1}^2 \leq \beta_{i_2}^2 \leq ... \leq \beta_{i_n}^2$ be the squared coefficients ordered in ascending order. We have

$$S_k = \sum_{j=n-k+1}^{n} \beta_{i_j}^2 = \sum_{\beta_i^2 \geq t_k^2} \beta_i^2,$$

where we assumed that the first retained coefficient $t_k := \beta_{i_{n-k+1}}$ is unique. If we consider t_k a fixed parameter instead of a random variable, the terms in the above sum are independent with expectation given by:

$$E[\beta_i^2 \mid \beta_i \geq t_k] = \frac{1}{1-\Phi(t_k)} \int_{t_k}^{+\infty} \frac{x^2 e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} dx,$$

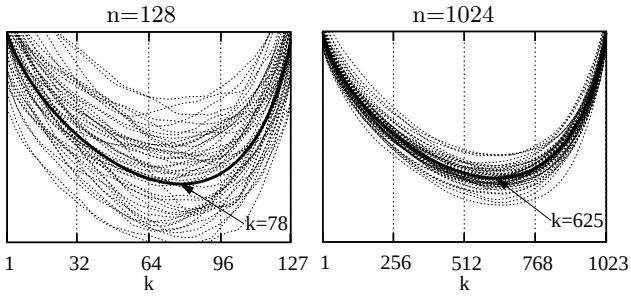


Figure 3: The renormalized NML denoising criterion with pure Gaussian noise.

where the expectation is taken with respect to the standard normal distribution whose distribution function is denoted by Φ. The integral is given by

$$\int \frac{x^2 e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} dx = \frac{-x e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} + \Phi(x),$$

and the expectation becomes

$$E[\beta_i^2 \mid \beta_i \geq t_k] = \frac{t_k e^{-\frac{t_k^2}{2}}}{\sqrt{2\pi}(1-\Phi(t_k))} + 1. \qquad (8)$$

Now in order to contain a k/n fraction of Gaussian random variates as n goes to infinity, the limiting value of the cut-point t_k must be

$$\lim_{n\to\infty} t_k = \Phi^{-1}\left(1 - \frac{k}{2n}\right).$$

(Division of k by two comes from the fact that also negative coefficients with large absolute value are included.) Plugging this into Eq. (8) in place of t_k gives the asymptotic behavior of the average S_k/k. Since the expectation of all coefficients under the unit variance Gaussian noise model is equal to one, the expectation of $(S_n - S_k)/(n-k)$, i.e., the expectation of the $n-k$ *smallest* squared coefficients can be easily obtained once the expectation of the k largest coefficients is known.

Fig. 3 shows the values of the renormalized NML denoising criterion with sample sizes $n = 128$ (on the left), and $n = 1024$ (on the right), with 50 repetitions in each case. Data is pure Gaussian noise with unit variance. The theoretical minima for the two samples sizes are $k = 78$ and $k = 625$ respectively. The asymptotic curve is plotted with a solid line. By evaluating the criterion for large n it can be seen that the MDL method tends to keep about $625/1024 \approx 61$ % of the coefficients. This is suboptimal in terms of both statistical risk and the natural meaning of information

and noise in data. If all data is indeed pure noise the method should indicate that there is no information in the data at all.

5.3 INTERPRETATION

Let us now consider the interpretation of the renormalized NML denoising criterion in order to understand the above described behavior. The code-length function (7) is the negative logarithm of a corresponding density of the following form (ignoring normalization constants):

$$(S - S_\gamma)^{-(n-k)/2} S_\gamma^{-k/2} = \|\hat{\beta}_{\gamma^c}\|^{-(n-k)} \|\hat{\beta}_\gamma\|^{-k}, \quad (9)$$

where γ^c denotes the complement of γ, i.e, the set of $n-k$ discarded coefficients.

Incidentally, the form in Eq. (9) is equivalent to using a zero-mean Gaussian density with optimized variance for both the retained and the discarded coefficients. This can be seen as follows. Given a vector x of k random variates, the maximal density achievable with a zero-mean Gaussian density assuming the entries in the vector are independent is given by

$$\max_\sigma (2\pi\sigma^2)^{-k/2} \exp\left(-\frac{\|x\|^2}{2\sigma^2}\right) = \left(2\pi e k^{-1} \|x\|^2\right)^{-k/2} \quad (10)$$

which is seen to be proportional to $\|x\|^{-k}$. Thus the two factors in Eq. (9) correspond to maximized Gaussian densities of the kind in (10). Fig. 4 gives an illustration verifying that the threshold is at the intersection points of two Gaussian densities fitted to the discarded and the retained coefficients respectively. The latter density has very high variance because the empirical distribution of the coefficients has heavy tails. The fact that both retained and discarded coefficients are encoded with a Gaussian density explains many aspects of the behavior reported above.

It is quite easy to derive rough conditions on when the criterion performs well. From orthogonality of the wavelet transform it follows that each of the informative coefficients is a sum of an information term and a noise term. Assuming independent noise, the density of the sum is given by the convolution of the densities of the summands. For instance, if the original signal has Gaussian density, the convolution is Gaussian as well with variance equal to the sum of the signal variance σ_S^2 and the noise variance σ_N^2. As long as the signal variance is large compared to the noise variance, the variance of the informative coefficients, $\sigma_S^2 + \sigma_N^2$, is significantly larger than that of the noise coefficients. Consequently, the criterion based on Gaussian densities with different variances is able to separate the informative and non-informative coefficients as long as the noise variance is not too high.[3] It is also easy to understand that fitting two Gaussian densities to a single one gives nonsensical results which explains the behavior in the pure noise scenario of Sec. 5.2.

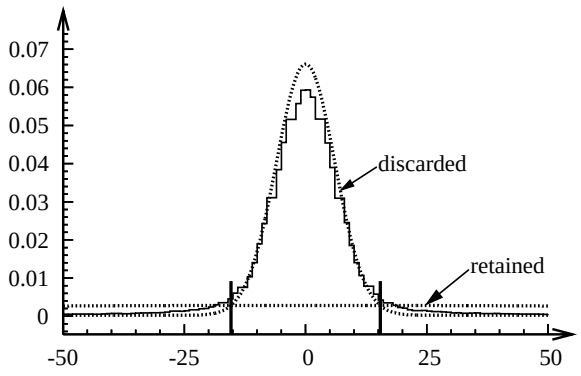


Figure 4: Gaussian densities fitted to noisy Lena ($\sigma = 10.0$). The empirical histogram is plotted with solid line. Gaussian densities with variance adjusted for the discarded ($\hat{\sigma} = 6.0$) and the retained ($\hat{\sigma} = 153.7$) coefficients are shown with dotted curves. Threshold is at ± 15.4.

It has been observed that wavelet coefficients in natural images tend to be well modeled by generalized Gaussian densities of the form $K \exp(-(|x|/\alpha)^\beta)$ where K is a normalization constant (Mallat, 1989). The typical values of β are near one which corresponds to the Laplacian (double exponential) density. This suggests that the density of the observed coefficients can be modeled by a convolution of the Laplace and Gaussian densities. Ruggeri & Vidakovic (1999) consider Bayes optimal hard thresholding in this model when the scale parameters of both densities are known. Chang *et al.* (2000) estimate the scale parameters from the observed signal. The construction of an NML model based on Laplacian and generalized Gaussian models with a proper treatment of the scale parameters is an interesting future research topic.

[3] Similar reasoning also shows that while the criterion is symmetric in the two sets of coefficients, one should always retain the k largest coefficients instead of the k *smallest* coefficients.

6 CONCLUSIONS

In its general form, the MDL principle form essentially aims at separating meaningful information from noise, and thus provides a very natural approach to denoising as an alternative to the statistical and Bayesian approaches. There are, however, some intricate issues in applying MDL to the denoising problem related to unbounded parametric complexity of Gaussian families. We discussed a solution by Rissanen involving a renormalization whose effect has been unclear so far and is of considerable interest not only in denoising applications but in the MDL framework in general.

The reported empirical and theoretical findings suggested a characterization of the domain of applicability for Rissanen's denoising method. It was seen that over-fitting is likely in the high noise regime. For practitioners, we pointed out two technical pitfalls and ways to avoid them. We gave an interpretation of the renormalization by showing that it results in a code based on two Gaussian densities, one for the retained wavelet coefficients and one for the discarded ones. Based on the interpretation we were able to explain both the empirical and the theoretical findings. The interpretation also facilitates understanding of the problem of unbounded parametric complexity in general and suggests generalizations of the renormalization procedure, potentially leading to improved MDL methods for denoising as well as other applications.

Acknowledgments

We thank Jorma Rissanen, Peter Grünwald, and Ursula Gather for useful discussions. This work was supported in part by the Academy of Finland under projects Minos and Cepler, the Finnish Technology Agency under project PMMA, and the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

Abramovich, F., Bailey, T., & Sapatinas, T. 2000. Wavelet analysis and its statistical applications. *Journal of the Royal Statistical Society, Series D*, **49**(1), 1–29.

Barron, A., Rissanen, J., & Yu, B. 1998. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, **44**(6), 2743–2760.

Chang, G., Yu, B., & Vetterli, M. 2000. Adaptive wavelet thresholding for image denoising and compression. *IEEE Transactions on Image Processing*, **9**(9), 1532–1546.

Daubechies, I. 1992. *Ten Lectures on Wavelets*. Society for Industrial & Applied Mathematics (SIAM), Philadelphia, PA.

Donoho, D., & Johnstone, I. 1991. *Minimax estimation via wavelet shrinkage*. Tech. rept., Stanford University.

Donoho, D., & Johnstone, I. 1994. Ideal spatial adaptation via wavelet shrinkage. *Biometrika*, **81**(3), 425–455.

Fodor, I.K., & Kamath, C. 2003. Denoising through wavelet shrinkage: an empirical study. *Journal of Electronic Imaging*, **12**(1), 151–160.

Foster, D.P., & Stine, R.A. 2001. The competitive complexity ratio. *Proceedings of the 2001 Conference on Information Sciences and Systems*, 1–6.

Foster, D.P., & Stine, R.A. 2005. The contribution of parameters to stochastic complexity. *To appear in:* Grünwald, P., Myung, I.J., & Pitt, M. (eds), *Advances in Minimum Description Length: Theory and Applications*, MIT Press, Cambridge, MA.

Grünwald, P. 2005. A Tutorial introduction to the minimum description length principle. *To appear in:* Grünwald, P., Myung, I.J., & Pitt, M. (eds), *Advances in Minimum Description Length: Theory and Applications*, MIT Press, Cambridge, MA.

Hansen, M., & Yu, B. 2000. Wavelet thresholding via MDL for natural images. *IEEE Transactions on Information Theory*, **46**(7), 1778–1788.

Mallat, S. 1989. A Theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11**(7), 674–693.

Ojanen, J., Miettinen, T., Heikkonen, J., & Rissanen, J. 2004. Robust denoising of electrophoresis and mass spectrometry signals with minimum description length principle. *FEBS Letters*, **570**(1–3), 107–113.

Rissanen, J. 1996. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, **42**(1), 40–47.

Rissanen, J. 2000. MDL denoising. *IEEE Transactions on Information Theory*, **46**(7), 2537–2543.

Ruggeri, F., & Vidakovic, B. 1999. A Bayesian decision theoretic approach to the choice of thresholding parameter. *Statistica Sinica*, **9**(1), 183–197.

Shtarkov, Yu M. 1987. Universal sequential coding of single messages. *Problems of Information Transmission*, **23**(3), 3–17.

Weaver, J.B., Yansun, X., Healy, D.M. Jr., & Cromwell, L.D. 1991. Filtering noise from images with wavelet transforms. *Magnetic Resonance in Medicine*, **21**(2), 288–295.

Focused Inference

Rómer E. Rosales†⁺ and Tommi S. Jaakkola†
†Computer Science and Artificial Intelligence Lab., MIT. Cambridge, MA 02139
⁺Computer-Aided Diagnosis and Therapy, Siemens Medical Solutions. Malvern, PA 19355*

Abstract

We develop a method similar to variable elimination for computing approximate marginals in graphical models. An underlying notion in this method is that it is not always necessary to compute marginals over all the variables in the graph, but focus on a few variables of interest. The Focused Inference (FI) algorithm introduced reduces the original distribution to a simpler one over the variables of interest. This is done in an iterative manner where in each step the operations are guided by (local) optimality properties. We exemplify various properties of the focused inference algorithm and compare it with other methods. Numerical simulation indicates that FI outperform competing methods.

1 INTRODUCTION AND RELATED WORK

Probabilistic models are useful in a wide variety of fields. An effective way to represent the structure of a probability distribution is by means of a graph; *e.g.,* a graphical model, where variables and their dependencies are associated to nodes and edges in the graph. A crucial task in using such models is to compute marginal distributions over single or groups of random variables. This is referred to here as probabilistic inference. However, the complexity of exact inference scales exponentially with the tree-width of the associated graphical model, and even finding -approximations (*i.e.,* within given error bounds) is NP-hard [2]. Approximate methods can nevertheless be indispensable in practice.

Approximate inference methods have relied on several key ideas. For example, we can try to simplify the original model to the extent that it becomes tractable. In some cases it is feasible to identify groups of nodes that are nearly conditionally independent or configurations that are highly improbable, and then modify the original graph appropriately to represent this finding before running an exact algorithm (*e.g.,* [9]). Variational methods, on the other hand, typically look for the best approximation within a restricted class of distributions, for example, by minimizing the KL-divergence $D(q||p)$ between the approximation q and the original distribution p[7]. The quality of this approximation is tied to how expressive the restricted class is. Other methods, such as Assumed Density Filtering (ADF)[13](see also [14]), Expectation Propagation (EP)[14] and sequential fitting[5] define the quality of approximation in terms of $D(p||q)$, preserving select statistics in the course of incorporating evidence. Similarly, belief Propagation (BP)[16, 12] and its generalizations[20, 19] seek locally (not globally) consistent beliefs about the values of variables and have been useful in various contexts.

Variational methods can generally provide a bound on the likelihood but are typically symmetry breaking in the sense that the optimized approximate marginals are asymmetric in the absence of any evidence to this effect (cf. mode selection). Propagation algorithms such as BP or EP avoid symmetry breaking due to the different optimization criterion. They are exact for trees, or hyper-trees in the case of generalized BP, but, with the singular exception of [18], do not provide bounds, nor are necessarily guaranteed to converge without additional assumptions.

The structure of the approximating distribution (*e.g.,* [6, 14, 15]), the message propagation scheme (*e.g.,* [19]), or the clusters in generalized BP can lead to important variability in accuracy; finding a *good* structure or clusters is an essential and still unresolved problem.

In this paper we pay closer attention to the essential operation for computing a subset of desired marginals,

*Current address. This work was done while the first author was with MIT CSAIL

i.e., marginalizing out each of the remaining hidden variables. The plain focused inference (FI) algorithm is a simple iterative process that eliminates variables step by step (or in parallel whenever possible) and obtains an approximation of the select marginal distribution. We extend the the basic FI idea to a distributed algorithm operating in a tree-like structure. Our method provides a formalism for performing the necessary graph/distribution transformation operations to be exact on restricted graphs, and approximate for others. While FI can be seen to generate approximating distributions at each step, these distributions do not have to be tractable.

2 DEFINITIONS - BACKGROUND

Let $X = (X_1, ..., X_N)$ be a random vector with X_i taking values denoted $\mathbf{x}_i$, where $\mathbf{x}_i \in \mathcal{X}$. We let $\mathcal{X}$ be the discrete space $\mathcal{X} = \{0, 1, ..., M-1\}$; thus, X takes values in the Cartesian product space $\mathcal{X}^N$. In this paper we consider probability distributions $p(\mathbf{x})$ whose structure is represented by the undirected bipartite graph $\mathcal{G} = (\{V, F\}; E)$, with variable nodes V such that $X = \{X_i | i \in V\}$, factor nodes F, and edges $E \quad \{(i, \ ) | i \in V, \quad \in F\}$. The factor graph [10] $\mathcal{G}$ corresponds to the following family of distributions:

$$p(X = \mathbf{x}) = \frac{1}{Z_p} \prod_{\alpha \in F} \psi_\alpha(\mathbf{x}_\alpha) \prod_{i \in V} \phi_i(\mathbf{x}_i), \qquad (1)$$

where ψ and ϕ are positive functions (potentials or factors), and X_α are all the random variables connected to the factor node $\ $; *i.e.,* $X_\alpha = \{X_i | (i, \ ) \in E\}$. For later convenience, we denote single node factors by ϕ. This graph representation is more explicit than the standard undirected graphical model representation regarding the factorization of the probability distribution. In this paper we concentrate primarily on the cases when p can be defined by factor nodes with degree at most two[1]. The neighborhood set of the variable node i is defined $\ (i) = \{j | (i, \ ) \in E, (j, \ ) \in E\}$ (this includes the node i itself), while the neighbors of the variable node i is the set $\ (i)^- = \ (i) - \{i\}$. The factors associated to a variable node i are denoted $F(i)$, with $F(i) = \{ \quad \in F | (i, \ ) \in E\}$. Throughout this paper, the short-hand $p(\mathbf{x})$ will denote $p(X = \mathbf{x})$.

3 FOCUSED INFERENCE APPROXIMATION

Consider the basic marginalization operation. When marginalizing the joint distribution $p(\mathbf{x})$ with respect to a single variable X_e, the fundamental computational issues, for discrete representations, are the time complexity of combining the relevant random variable configurations and the space complexity of representing the result. In general we have that with $\bar{e} = V - \{e\}$,

$$\sum_{\mathbf{x}_e} p(\mathbf{x}) \propto h_2(\mathbf{x}_{\bar{e}}) \sum_{\mathbf{x}_e} h_1(\mathbf{x}_{\nu(e)}) = h_2(\mathbf{x}_{\bar{e}}) f(\mathbf{x}_{\bar{e}}). \qquad (2)$$

Even if the graph corresponding to p is a tree, representing $f(\mathbf{x}_{\bar{e}})$ without resorting to its functional form may require $\mathcal{O}(M^{|\nu(e)|-1})$ space. Further computations (like marginalizing with respect to another variable), referring to this result may also have exponential time complexity.

This exact operation can been seen as a step in a bucket elimination algorithm [3]. This is also the basic operation that data structures like the clique-tree or junction tree are designed to handle in exact inference methods like variable elimination [17], and illustrates why some elimination and induced triangulations are much more efficient than others, even though all perform exact calculations.

There are instances when f turns out to accept simple (*e.g.,* product) decompositions. In this case it is possible to improve upon the above complexity bounds on inference. However, it is not clear how to induce such decompositions given a distribution p. The essence of the focused inference algorithm explained in this section lies in variable elimination and in generating a succession of non-exact decompositions to compute optimal marginal approximations in the context of Eq. 2. We will discuss an extension of the basic algorithm later in the paper.

3.1 BASIC FI ALGORITHM

Let $p(\mathbf{x})$ be the distribution of interest, with associated factor graph $\mathcal{G} = (\{V, F\}, E)$, focused inference (FI) is based on a new graph decomposition together with an approximation that reduces the original distribution p (and graph $\mathcal{G}$) to a *simpler* one in an iterative manner, with certain optimality properties at each step. We can think of the essential process as *focusing* only on a few target node(s) at a time, whose marginal distributions (*e.g.,* pairwise) are to be approximated. Each iteration eliminates variable and factors, includes new factors, and modifies the distribution appropriately to keep the focused approximation accurate.

We start by formalizing FI for a single iteration and when the target variables consist of a specific pair of nodes $\mathcal{T}$, $\mathcal{T} \subset V$, and later generalize it to multiple pairwise marginals.

The first step of the iteration consists on choosing a non-target node $e \in V - \mathcal{T}$ and rewriting the corre-

[1]Note this need not be the case for joint marginals of p

sponding probability distribution as:

$$p(\mathbf{x}) = \tilde{p}_1(\mathbf{x}_{\nu(e)})\tilde{p}_2(\mathbf{x}_{\bar{e}}), \quad (3)$$

where the two new distributions are defined according to the following decomposition:

$$\tilde{p}_1(\mathbf{x}_{\nu(e)}) = \frac{1}{Z_{\tilde{p}_1}} \prod_{\alpha \in F(e)} \psi(\mathbf{x}_\alpha) \prod_{i \in \nu(e)} \phi_i(\mathbf{x}_i) \quad (4)$$

$$\tilde{p}_2(\mathbf{x}_{\bar{e}}) \propto \prod_{\alpha \notin F(e)} \psi(\mathbf{x}_\alpha) \prod_{i \notin \nu(e)} \phi_i(\mathbf{x}_i). \quad (5)$$

Assuming each factor involves at most two variables, $\tilde{p}_1(\mathbf{x}_{\nu(e)})$ is always a tree-structured distribution and thus computing exact marginals from $\tilde{p}_1$ can be done efficiently. $\tilde{p}_2(\mathbf{x}_{\bar{e}})$ remains generally intractable. The decomposition is not unique (even up to constant) since we are free to trade single node marginals between the components. Note that the decomposition itself involves no approximations, only rewriting of the original distribution.

The exact node/edge removal operation (marginalizing) consist on finding $f(\mathbf{x}_{\nu(e)^-}) = \sum_{\mathbf{x}_e} \tilde{p}_1(\mathbf{x}_{\nu(e)})$; see Fig. 1(b). The above decomposition is helpful since sensible approximations for the first portion of the full distribution (Eq. 4) are readily available. In particular, in this paper we employ the specific class of approximations that optimize the KL-divergence $D(f(\mathbf{x}_{\nu(e)^-})||q(\mathbf{x}_{\nu(e)^-}))$ between $f(\mathbf{x}_{\nu(e)^-})$ and the approximating distribution $q(\mathbf{x}_{\nu(e)^-})$ constrained to be a tree-distribution. We denote this class of approximating distributions by Q.

In the second step of the FI iteration we solve:

$$q(\mathbf{x}_{\nu(e)^-}) = \arg\min_{q \in Q} D(\sum_{\mathbf{x}_e} \tilde{p}_1(\mathbf{x}_{\nu(e)})||q(\mathbf{x}_{\nu(e)^-}). \quad (6)$$

This projection can be solved efficiently whenever Q is restricted to trees.

This optimization is related to that used by ADF (and thus EP); however in FI no fixed, predetermined reference (*e.g.,* tree-structured) distribution is set and at each step the structure of the best local approximating distribution can be obtained dynamically. Also, the projection operation may (automatically) introduce factors that were not previously present. We are not assuming a specific choice of ADF *terms*. Also, recall than in ADF the structure of the approximating distribution is predetermined (not found by ADF itself).

One way to represent the solution to Eq. 6 is by the following tree-structured factorization:

$$q(\mathbf{x}_{\nu(e)^-}) = \prod_{(i,j) \in E_T} q(\mathbf{x}_i, \mathbf{x}_j) / \prod_{i \in V_T} q(\mathbf{x}_i)^{d_i - 1}, \quad (7)$$

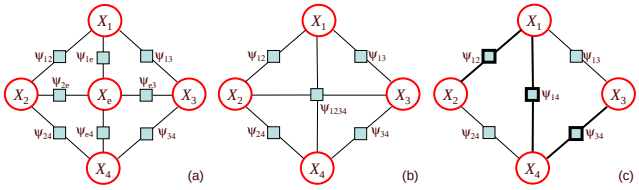


Figure 1: One-step approximation by removal of X_e: (a) original factor graph, (b) exact (marginalized) graph, and (c) example FI approximation where appropriate factors (in bold) have been created (ψ_{14}) and updated (ψ_{12}, ψ_{34})

where d_i denotes the degree of node i and $\mathcal{G}_T = (V_T, E_T)$ is a tree (q is in the family of distributions Q). The optimal tree $\mathcal{G}_T$ can be found efficiently.

In the third step of the algorithm we combine the projected approximation with the remaining variables to get the approximation to marginalized $p(\mathbf{x})$:

$$\hat{p}(\mathbf{x}_{\bar{e}}) = q(\mathbf{x}_{\nu(e)^-})\tilde{p}_2(\mathbf{x}_{\bar{e}}). \quad (8)$$

This node/edge elimination and approximation iteration is repeated until all but the focus set $\mathcal{T}$ is left in the graph [2]. The new distribution $\hat{p}(\mathbf{x}_{\bar{e}})$ is again defined in the form of Eq. 1. This involves redefining the previous potentials and incorporating new ones. The following provides the resulting factor/potential update equations [3]. For each pair $(i,j) \in E_T$, potentials can be modified or created:

$$\psi_\alpha(\mathbf{x}_\alpha) \leftarrow \psi_\alpha(\mathbf{x}_\alpha) \frac{q(\mathbf{x}_i, \mathbf{x}_j) q(\mathbf{x}_i)^{-\rho_i} q(\mathbf{x}_j)^{-\rho_j}}{\phi_i(\mathbf{x}_i)\phi_j(\mathbf{x}_j)} \text{ (modify)} \quad (9)$$

$$\psi_\alpha(\mathbf{x}_\alpha) \leftarrow \frac{q(\mathbf{x}_i, \mathbf{x}_j) q(\mathbf{x}_i)^{-\rho_i} q(\mathbf{x}_j)^{-\rho_j}}{\phi_i(\mathbf{x}_i)\phi_j(\mathbf{x}_j)} \text{ (create)}, \quad (10)$$

where $_i = (d_i - 1)/d_i$ and the potential is modified whenever both $(i, \ )$ and $(j, \ )$ are in E (created otherwise). The graphical operations for factor graph $\mathcal{G}$ are variable node, factor, edge removal, and edge addition, respectively:

$$(i) \quad V \leftarrow V - \{e\}, \quad (ii) \quad F \leftarrow F - F(e),$$
$$(iii) \quad E \leftarrow E - \{(e, \ )| \ \in F(e)\},$$
$$(iv) \quad E \leftarrow E \cup \{(i, \ ), (j, \ )|(i,j) \in E_T\}$$

One iteration applied to a simple five-variable factor graph is shown in Fig. 1. The repetition of these steps defines an elimination ordering $\mathcal{E} = (e_1, ..., e_K)$ and a series of approximating distributions $\{\mathcal{Q}_k\}_{k=1,...,K}$ which characterize one focusing operation. While these basic steps are fixed, the global algorithm is more flexible; for example, in the choice of approximating distributions, in the elimination ordering, etc. These and other aspects will be further discussed next.

[2] Alternatively until a tractable substructure containing the set $\mathcal{T}$ has been reached

[3] This is one succinct way to state the update equations for multinomials, other equivalent forms can be also used. This form does not require updating the potentials ϕ_i

4 ALGORITHM ANALYSIS

Here we illustrate key properties of the algorithm and provide additional details.

4.1 ALGORITHM COMPLEXITY AND OPTIMALITY

Under the decomposition defined in Eqs. 4-5, the minimization problem in Eq. 6 for a fixed tree structure has a known solution in $\mathcal{O}(M^3)$. The problem reduces to finding the pairwise marginals of $\tilde{p}_1$ along the tree edges E_T. Since $\tilde{p}_1$ is always a tree (a star graph centered at X_e) and each potential ψ is a function of at most two random variables, any of these marginals can be found in $\mathcal{O}(M^3)$. As for finding the best tree-structured distribution family in Q, this result follows directly from [1] applied to our decomposition.

Proposition 1 *The focused inference algorithm of Sec. 3 is exact for any decimatable distribution p (tree-width two or, equivalently, when the maximal clique size of the triangulated graph is three). Not all elimination orderings yield the exact result.*

Proof The approximation is exact when all the steps are exact. The steps are exact if each variable has at most two neighbors when eliminated. Since the maximal clique size is three, this elimination constraint can always be satisfied through some elimination ordering.

An analogous result may not hold for ADF (or EP) with the same time complexity.

4.2 ELIMINATION ORDERING

For a graph $\mathcal{G} = (\{V, F\}, E)$ and for a set of nodes $\mathcal{F}$ of interest, an elimination ordering is a sequence of nodes $\mathcal{E} = (e_1, ..., e_K)$ with $e_i \in V - \mathcal{F}$. In case we measure the approximation accuracy in terms of the KL-divergence, the focused inference method suggests a seemingly natural elimination ordering $\mathcal{E}$: at each step, eliminate the (non-target) node that gives the lowest KL divergence $D(f(\mathbf{x}_{\nu(e)^-})||q(\mathbf{x}_{\nu(e)^-}))$ between marginalized and approximating distribution at each step. However, note that this gives a locally best and, in general, not a globally best ordering. Finding approximations to the best overall elimination ordering is a much harder problem due that the complexity of the problem representation increases rapidly with the number of elimination steps.

The focused inference algorithm is designed to concentrate on specific marginals and thus, a particular ordering is used to reduce the graph to those nodes of interest. In the most general setting, the algorithm has to be *run* again if other marginals are needed (or partially run since common calculations or partial results could be handily stored). A reasonable question is whether these marginals are consistent. The answer is negative, in general. Specifically, for a distribution $p(\mathbf{x})$ and two sets of focus (target) variables $\{X_a, X_b\}$ and $\{X_b, X_c\}$, the corresponding pairwise and single marginals produced by the focused inference algorithm under different elimination orderings $\mathcal{E}_1 = (e_{11}, ..., e_{1K_1})$ and $\mathcal{E}_2 = (e_{21}, ..., e_{2K_2})$ are consistent (1)if $p(\mathbf{x})$ is a decimatable distribution; since the algorithm is exact, and (2)if the elimination orderings are the same except for the final node: $K_1 = K_2 = N - 2$ and $e_{1i} = e_{2i}$ for $i < K$; since after eliminating $N - 3$ nodes, the remaining variables will be X_a, X_b, X_c, and their joint distribution is decimatable.

If we do not require that the approximation be optimal (at each step), the graph can be reduced to a tree very quickly. Moreover, this can be done so that the pairwise distributions are tree consistent. However, clearly this involves non-optimal (local) approximations.

4.3 CONSISTENCY OF SINGLE AND PAIRWISE MARGINALS

Let us instead consider how to start from the potentially inconsistent set of marginals found using the FI algorithm and reach a consistent set of marginals. We consider the following problem: given a set of pairwise marginals found under different elimination orderings, how can we obtain a set of consistent marginals with respect to a tree graph $\mathcal{G}_{T'} = (V_T, E_{T'})$.

Our approach consist on using a maximum likelihood criterion; specifically, let $\mathcal{M}$ be the set of ordered pairs (i, j) of marginals $\{q(\mathbf{x}_i, \mathbf{x}_j)\}$. Under this criterion, we wish to solve the following optimization problem:

$$\arg\max_{r \in R} - \sum_{(i,j) \in \mathcal{M}} q(\mathbf{x}_i, \mathbf{x}_j) \log r(\mathbf{x}_i, \mathbf{x}_j), \qquad (11)$$

for the set of distributions R with model structure $\mathcal{G}_{T'}$. This problem is equivalent to the minimization of Eq. 6, and thus can be solved in closed-form.

4.4 INCLUDING SINGLE NODE POTENTIALS

The decomposition given by Eqs. 4- 5 is an instance of a more general decomposition which generalizes the way single node potentials are included as follows:

$$\tilde{p}_1(\mathbf{x}_{\nu(e)}) \propto \phi_e(\mathbf{x}_e) \prod_{\alpha \in F(e)} \psi(\mathbf{x}_\alpha) \prod_{i \in \nu(e)^-} \phi_i(\mathbf{x}_i)^{\eta_i} \qquad (12)$$

$$\tilde{p}_2(\mathbf{x}_{\bar{e}}) \propto \prod_{\alpha \notin F(e)} \psi(\mathbf{x}_\alpha) \prod_{i \notin \nu(e)} \phi_i(\mathbf{x}_i) \prod_{i \in \nu(e)^-} \phi_i(\mathbf{x}_i)^{(1-\eta_i)} \qquad (13)$$

which subsumes the original one.

The extra degrees of freedom are given by the variables $\eta = \{\eta_i\}, i \in \ (e)^-, \ \eta_i \in \Re$ (note that $\phi_e(\mathbf{x}_e)$ must be fully included during e's elimination). This extra flexibility could be used to our advantage in finding a better distribution when minimizing Eq. 6. This is because single node potentials could be included such as to obtain an approximating q distribution giving smaller KL-divergence. However, one must be cautious, since we can almost always define η to obtain an approximating distribution for which the KL-divergence is almost zero. This can be done by allowing almost all of p's probability mass to fall in appropriate variable states easily represented by distributions in Q. Yet, despite this momentary success, the overall gain is not guaranteed to be larger since the resulting $\hat{p}$ distribution might be difficult to approximate in future steps. This may allow us to provide more accurate overall approximations by appropriately including the effect of single node potentials. Taking advantage of this generalization is an interesting problem that remain to be exploited.

4.5 FURTHER CONNECTIONS WITH OTHER METHODS

As a way to further understand FI, we now discuss other connections with related methods. Consider a single inclusion of a pairwise term in ADF. The marginals obtained by ADF for any (tractable) approximating distribution could also be obtained by FI. This can be seen by noticing that FI can perform exact marginalization in a cycle (like ADF) and the inclusion of a pairwise term in ADF will at most generate a cycle. It is significant that for this equivalence to hold, FI needs to make locally suboptimal decisions and ignore those edges not in the cycle, even if they can be easily approximated during elimination. For simultaneous inclusion of multiple terms, ADF's complexity in general increases exponentially, FI's complexity remains as before, of course using an approximation.

Unlike ADF and EP, in FI there is no reference structure for the approximation made. Interestingly, intermediate (joint) approximations built by FI may not be tractable. Their structure can be chosen with locally optimal guarantees. FI builds these approximations dynamically, thus there are less choices to be made by hand regarding the structure of the approximating distribution. This is related to [5], in the sense that different approximating structures are found at each step; however, in [5] a (tractable) tree-structured joint distribution is maintained at all times.

There exist certain resemblance between the FI algorithm in Sec. 3.1 and the mini-buckets scheme [4] in the sense that both methods repeatedly approximate complex functions of multiple variables with products of simpler functions. In mini-buckets, the local approximations employ a non explicitly guided partitioning of variables to functions (a partitioning, to some extent corresponds to the structure of a local approximating distribution in the FI method). In FI, contrary to mini-buckets, the approximating distributions q, including their structure, can be solved for, and are optimal with respect to a definite criterion, the appropriate KL-divergence.

In mini-buckets the approximation relies on arithmetic bounds on product of positive functions. In a criterion derived from mini-buckets [11], this approximation is given by the solution of a linear optimization problem (thus more like FI). However, still the structure of the approximating distribution is not part of the formulation and also the optimization problem entails using an exponential number of constraints.

5 DISTRIBUTED FOCUSED INFERENCE

Let us say we are interested in the marginal distribution for all of the variables X_i. In the worst case, the basic FI algorithm needs to be re-run on the order of N times to obtain all marginals of interest. Is there a more efficient way to perform the necessary computations? In this section we address the question whether there exist a distributed (asynchronous) algorithm equivalent or based on the same fundamental ideas. For exact inference, there are distributed algorithms to some extent related to variable elimination (*e.g.,* [17, 16, 12]). Asynchronous algorithms also exist for fixed structure approximations (*e.g.,* [13, 7, 14]). However, note that in the FI approximation, the result of eliminating one variable is not propagated symmetrically through the graph (neighbors), it depends on the factorization chosen; the underlying factor graph is dynamically modified, based on previous approximations to other parts of the graph; and different elimination orderings (optimal in some sense for a particular focusing variable) are used for computing the different marginals. Thus, it is not clear for example, what data structure fits the underlying algorithm, what information or quantities a node should transfer to another, and if the overall algorithm allows for quantities to be stored efficiently, *e.g.,* locally.

It turns out that for a type of FI approximations, we can build a distributed algorithm and answer these questions. In order to define the algorithm, a tree structure similar to the clique-tree [12, 17] can be used for the underlying computations. However, unlike the above, in our case it is not necessary to find the maximal cliques or use the concept of triangulation explicitly. This is important because finding maximal cliques

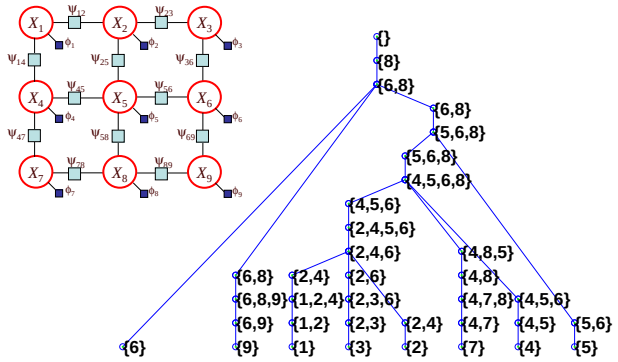


Figure 2: Example factor graph and the tree induced by the ordering $\mathcal{E} = (1, 3, 7, 9, 2, 4, 5, 6, 8)$

is in NP-complete [8]. The following algorithm defines the necessary tree structure whose nodes, denoted $\mathcal{A}_j$, are subsets of the random variables in p:

Algorithm for building Order-induced Tree

Denote elimination ordering $\mathcal{E} = (e_1, ..., e_n)$

1. Assign a single variable to the initial tree nodes: $\mathcal{A}_j = \{X_{e_j}\}$ $(j = 1, ..., N-1)$
2. Iterate $i = 1...N$
   (a) Create new node $\mathcal{C} = \bigcup_j \mathcal{A}_j$ for j s.t. $\mathcal{A}_j$ does not have a parent **and** $X_{e_i} \in \mathcal{A}_j$
   (b) For each j found in (a)
   Let $\mathcal{B} = \mathcal{C} - \mathcal{A}_j = \{X_{b_l}\}$ and chain nodes: $(\mathcal{C}) \rightarrow (\mathcal{C} \backslash \{X_{b_1}\}) \rightarrow ... \rightarrow (\mathcal{C} \backslash \mathcal{B}) \rightarrow (\mathcal{A}_j)$
   (c) For all the (not eliminated) variables X_k sharing a factor with X_{e_i}
      i. Create new node $\mathcal{C}' = \mathcal{C} \cup X_k$ and make $\mathcal{C}'$ a parent of $\mathcal{C}$
      ii. Redefine $\mathcal{C} \leftarrow \mathcal{C}'$
   (d) Create new node $\mathcal{C}' = \mathcal{C} - \{X_{e_i}\}$ and make $\mathcal{C}'$ a parent of $\mathcal{C}$
   (e) Eliminate X_{e_i}

Since we do not need the concept of cliques, we simply call it an *Order-induced Tree* (OT). Note that when traversed bottom-up (to the root), this tree gives a marginalization ordering that properly tracks the resulting function arguments at the nodes in the order given. As in the basic FI, the above steps resemble bucket elimination [3]. In fact, at the graph level, the OT algorithm performs the variable inclusion and elimination operations in the same order as FI. An example OT tree for a simple 3×3 grid is shown in Fig. 2. The distributed algorithm (shown next) uses the OT as basic data structure for message passing.

Distributed Focused Inference Algorithm

1. Form OT and for each node $\mathcal{A}$ assign function:

$$\tilde{\psi}_{\mathcal{A}}(\mathbf{x}_{\mathcal{A}}) = \prod_{\alpha \in F(\mathcal{A})} \psi_{\alpha}(\mathbf{x}_{\alpha}) \prod_{i \in \mathcal{A}} \phi_i(\mathbf{x}_i), \quad (14)$$

   $F(\mathcal{A})$: set of factors whose variables are in $\mathcal{A}$

2. Pick any node in the tree as root node
3. Perform a bottom-up and top-down pass, sending the following message between neighbor nodes (random variable sets) $\mathcal{A}$ and $\mathcal{B}$:

$$m_{\mathcal{B} \rightarrow \mathcal{A}}(\mathbf{x}_{\mathcal{A}}) = \wp_{\mathbf{x}_{\mathcal{B} \backslash \mathcal{A}}}[\prod_{\mathcal{C} \in \nu(\mathcal{B})} \frac{m_{\mathcal{C} \rightarrow \mathcal{B}}(\mathbf{x}_{\mathcal{B}})}{\tilde{\psi}_{\mathcal{C} \cap \mathcal{B}}(\mathbf{x}_{\mathcal{C} \cap \mathcal{B}})} \tilde{\psi}_{\mathcal{B}}(\mathbf{x}_{\mathcal{B}})], \quad (15)$$

   where denotes neighborhood in the OT

4. Compute marginals for the nodes of interest:

$$p(\mathbf{x}_{\mathcal{A}}) \propto \prod_{\mathcal{B} \in \nu(\mathcal{A})} \frac{m_{\mathcal{B} \rightarrow \mathcal{A}}(\mathbf{x}_{\mathcal{A}})}{\tilde{\psi}_{\mathcal{B} \cap \mathcal{A}}(\mathbf{x}_{\mathcal{B} \cap \mathcal{A}})} \tilde{\psi}_{\mathcal{A}}(\mathbf{x}_{\mathcal{A}}) \quad (16)$$

   (*e.g.,* use the nodes $\mathcal{A}$ containing the single variables of interest; some joint marginals can be computed directly as well)

In the algorithm, the operator $\wp$, has a similar role than the minimization in Eq. 6. In the case of FI we have:

$$\wp_{\mathbf{x}_{\mathcal{B} \backslash \mathcal{A}}}[g] \triangleq \arg\min_{q \in \mathcal{Q}} D(\frac{1}{Z_g} \sum_{\mathbf{x}_{\mathcal{B} \backslash \mathcal{A}}} g(\mathbf{x}_{\mathcal{B}}) || q(\mathbf{x}_{\mathcal{A}})), \quad (17)$$

which is the known projection operation in information geometry (as before $\mathcal{Q}$ is the set of tree structure-distributions). Since D is defined on distributions, Z_g is the necessary normalization constant. We defer a detailed analysis of the above algorithm and present the basic results in the remaining of this section.

Theorem 2 *The distributed algorithm computes the exact marginals when the minimization operation is replaced by exact summation, i.e., when* $\wp_{\mathbf{x}_{\mathcal{B} \backslash \mathcal{A}}}[g] \triangleq \frac{1}{Z_g} \sum_{\mathbf{x}_{\mathcal{B} \backslash \mathcal{A}}} g(\mathbf{x}_{\mathcal{B}})$

Proof sketch It suffices to show that the new definition of $\wp$ is equivalent to solving for f in Eq. 2, and that sequential application of step 3 with any root node is equivalent to sequential application of Eq. 2 in a particular ordering.

From the above result, step 3 using Eq. 17 can be seen as passing (approximate) distributions over variables in the target nodes. Interestingly these distributions may be intractable themselves. The basic FI

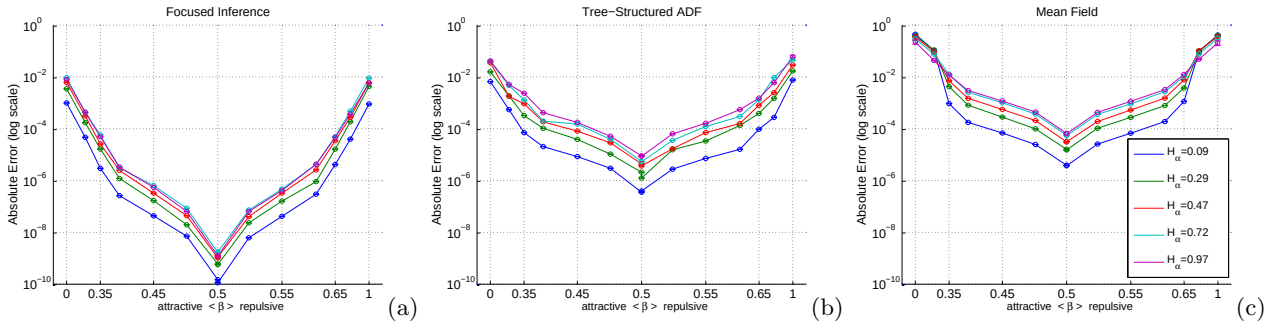


Figure 3: Numerical test results for grid networks with random single and pairwise potentials with various levels of entropy bounds (H_α,I_β). Note x-axis scale is given in terms of I_β and networks with maximally attractive and repulsive potentials are at $\beta = 0$ and $\beta = 1$ respectively. Performance of: (a) FI, (b) ADF, and (c) MF

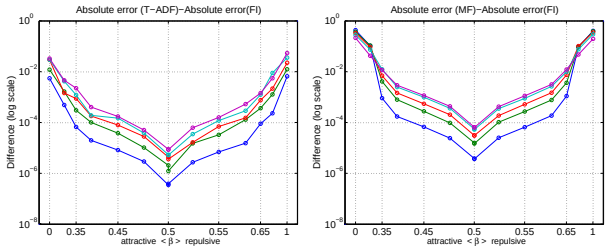


Figure 4: Absolute error differences (see Fig. 3 for legend)

algorithm, focusing on a single marginal, is equivalent to (1)choosing an OT with same ordering $\mathcal{E}$ and (2)performing just one pass (to the root). However, the marginals computed by running the basic FI algorithm for multiple focusing sets are not necessarily the same as those computed by the distributed algorithm. This can be easily seen by observing that different approximations are made in each case. The distributed algorithm is equivalent to multiple runs of FI where every run respects the approximations induced by the OT by means of its variable subsets and tree arrangement. The approximation over the target variables can vary in structure (*i.e.,* we can still choose optimal tree-structure distributions for message passing).

6 NUMERICAL EVALUATION

In order to test how FI performs for diverse types of distributions, we constructed a number of binary 9×9 grid networks by choosing its factors $\psi_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \theta_{\mathbf{x}_i\mathbf{x}_j}$ according to different uniform priors. Specifically, we use a hyper parameter to define the random variable $b \sim \mathcal{U}(\frac{1}{2}, 1 -)$ and set $\theta = (\theta_{00}, \theta_{01}; \theta_{10}, \theta_{11}) = (\frac{b}{2}, \frac{1-b}{2}; \frac{1-b}{2}, \frac{b}{2})$. When letting $0 < < \frac{1}{2}$, attractive potentials with varying strength are constructed. Similarly, by letting $\frac{1}{2} < < 1$ and $b \sim \mathcal{U}(1 - , \frac{1}{2})$ we define repulsive potentials. By varying we control the (maximum allowed) mutual information I_β describing how dependent the states of node pairs are: maximum dependency is achieved at $= 0$ (attractive) and $= 1$ (repulsive), and full independency at $= \frac{1}{2}$.

A second hyper parameter controls the distribution of single node potentials. Let $\phi_i(\mathbf{x}_i) = \theta_{\mathbf{x}_i}$, we define $a \sim \mathcal{U}(\frac{1}{2} - , \frac{1}{2} +)$, for $0 < < \frac{1}{2}$ and let $(\theta_0, \theta_1) = (a, 1 - a)$. Thus, controls the entropy in the prior state of a single variable (associated to its single node potential); when $= \frac{1}{2}$ the minimum entropy allowed, denoted H_α, is 1 bit (full uncertainty), and when $= 0$ it is 0 bits. In our experiments we varied and to obtain probability distributions with different properties regarding strength of dependences and strength of value preference.

In all of the experiments, we used the basic FI algorithm (no consistency was enforced). We chose the node to be eliminated at each step simply by looking at the number of neighbors of each node in the intermediate graphs and picking those with less neighbors first, randomly when tied. For ADF, we chose the structure of the approximating distribution by keeping the most informative edges (maximizing the pairwise mutual information) forming a spanning tree. This criterion performed better than random edge selection.

Fig. 3(a) shows the accuracy of FI for probability distributions sampled under different settings of the hyper-parameters and . Performance is measured in terms of the average absolute difference between exact and approximate (single node) marginals. As expected the performance improves as the coupling between the nodes become weaker ($\rightarrow \frac{1}{2}$) for all values of . We performed the same tests using ADF and the variational Mean Field method. Fig. 3(b)(c) shows the performance results from these methods. Like FI, accuracy is higher at $\frac{1}{2}$ for any .

FI clearly outperforms ADF under all conditions (Fig. 4-left). The ADF *terms* were the pairwise factors; FI and ADF had equivalent computational complexity.

Notably, the difference in performance increased with the strength of the dependences in the network and also with the strength of the *field* given by the single node potentials. Focused inference also outperformed Mean Field (MF) under all conditions (Fig. 4-right), even in the case when the basic Mean Field assumption is almost fully valid (when variables are almost independent). As variable dependencies grew stronger, the performance gap between FI and the competing methods became larger at increasing rates.

7 CONCLUSIONS

We introduced an approximate inference algorithm, similar to variable elimination, that is based on tailoring the approximation to the subset of variables of interest. We also developed a distributed message-passing version of the algorithm, constructed around a particular elimination ordering.

The basic decomposition step, followed by the projection, can be guaranteed to be optimal for decimatable graphs and properly chosen elimination ordering. We are not aware of similar results for ADF. In a more general context, the advantage of the focused inference algorithm lies primarily in the inclusion of dependencies induced by marginalization but not represented in the original graph. FI does not require setting a fixed reference distribution (*e.g.,* a class of tractable approximating distributions) for defining the approximation. The selection of dependencies to introduce is based on optimizing the projection of each local marginalization result down to a tree. The ability to maintain such dependencies through approximate marginalizations may underlie the superior empirical results.

Acknowledgements

The authors gratefully acknowledge support from DARPA IPTO program and British Aerospace Engineering.

References

[1] C. Chow and C. Liu, *Approximating discrete probability distributions with dependence trees*, Trans. Information Theory. **14** (1968).

[2] P. Dagum and M. Luby, *Approximating probabilistic inference in bayesian belief networks is NP-hard*, Artificial Intelligence **60** (1993), no. 1, 141–153.

[3] R. Dechter, *Bucket elimination: A unifying framework for reasoning*, Artificial Intelligence **113** (1999), no. 1-2, 41–85.

[4] R. Dechter and I. Rish, *Mini-buckets: A general scheme for approximating inference*, J. ACM **50** (2003), no. 2, 107–153.

[5] B. J. Frey, R. Patrascu, T. Jaakkola, and J. Moran, *Sequentially fitting inclusive trees for inference in noisy-or networks*, Neural Info. Proc. Systems, 2000.

[6] Z. Ghahramani and M. Jordan, *Factorial hidden markov models*, Neural Info. Proc. Systems, 1997.

[7] M. Jordan, Z. Ghahramani, T.Jaakkola, and L. Saul, *An introduction to variational methods for graphical models*, Learning in Graphical Models, M. Jordan (ed.) (1998).

[8] R. M. Karp, *Reducibility among combinatorial problems*, In R. E. Miller and J. W. Thatcher, eds., Complexity of Computer Computations (1972), 85–104.

[9] U. Kjærulff, *Reduction of computational complexity in Bayesian networks through removal of weak dependencies*, Proc. Uncert. in Artif. Intell., 1994.

[10] F. Kschischang and B. Frey, *Iterative decoding of compound codes by probability propagation in graphical models*, J. Sel. Areas in Comm. **16** (1998), 219–230.

[11] D. Larkin, *Approximate decomposition: A method for bounding and estimating probabilistic and deterministic queries*, Proc. Uncert. in Artif. Intell., 2003.

[12] S. L. Lauritzen and D. J. Spiegelhalter, *Local computations with probabilities on graphical structures and their application to expert systems*, J. Royal Stat. Society, B **50** (1988), no. 2, 157–223.

[13] P. Maybeck, *Stochastic models estimation and control*, Academic Press, 1982.

[14] T. Minka, *A family of algorithms for approximate Bayesian inference*, Ph.D. thesis, MIT, 2001.

[15] T. Minka and Y. Qi, *Tree-structured approximations by expectation propagation*, Neural Info. Proc. Systems, 2003.

[16] J. Pearl, *Probabilistic reasoning in intelligent systems*, Morgan-Kaufman, 1988.

[17] G. Shafer and P. Shenoy, *Probability propagation*, Ann. Math. Artificial Intel. **2** (1990), 327–351.

[18] M. Wainwright, T. Jaakkola, and A. Willsky, *A new class of upper bounds on the log partition function*, Proc. Uncert. in Artif. Intell., 2002.

[19] ———, *Tree-based reparameterization framework for analysis of sum-product and related algorithms*, Trans. Information Theory. **49-5** (2003).

[20] J. Yedidia, W. Freeman, and Y. Weiss, *Generalized belief propagation*, Neural Info. Proc. Systems, 2000, pp. 689–695.

Kernel Methods for Missing Variables

Alex J. Smola, S.V.N. Vishwanathan
Statistical Machine Learning Program
NICTA and ANU, Canberra, ACT, 0200
{Alex.Smola, SVN.Vishwanathan}@nicta.com.au

Thomas Hofmann
Department of Computer Science
Brown University, Providence, RI
th@cs.brown.edu

Abstract

We present methods for dealing with missing variables in the context of Gaussian Processes and Support Vector Machines. This solves an important problem which has largely been ignored by kernel methods: How to systematically deal with incomplete data? Our method can also be applied to problems with partially observed labels as well as to the transductive setting where we view the labels as missing data.

Our approach relies on casting kernel methods as an estimation problem in exponential families. Hence, estimation with missing variables becomes a problem of computing marginal distributions, and finding efficient optimization methods. To that extent we propose an optimization scheme which extends the Concave Convex Procedure (CCP) of Yuille and Rangarajan, and present a simplified and intuitive proof of its convergence. We show how our algorithm can be specialized to various cases in order to efficiently solve the optimization problems that arise. Encouraging preliminary experimental results on the USPS dataset are also presented.

1 Introduction

Kernel methods [12] have been remarkably successful for standard classification and regression problems. However, they have also been found very effective in dealing with a variety of related learning problems such as sequence annotation, conditional random fields, multi-instance learning, and novelty detection. Many algorithms for Gaussian Processes (GP) and Support Vector Machines (SVM) bear witness of this. One problem, however, has remained completely untouched so far: How to deal with datasets which exhibit missing variables?

In the following, we will develop a framework to deal with such cases in a systematic fashion. Our analysis is based on the observation that kernel methods can be written as estimators in an exponential family. More specifically, Gaussian Processes can be seen to be minimizing the negative log-posterior under a normal prior on the natural parameter of the exponential density, whereas Support Vector Machines maximize the likelihood ratio. Based on this observation, we provide a method for dealing with missing variables in such a way that standard kernel methods arise as a special case, whenever there are no missing variables.

To solve the optimization problems arising in this context – a concave-convex objective function with both convex and concave constraints – we extend the CCP algorithm of [16] for finding local optima and give an intuitive proof for its convergence.

The rest of the paper is organized as follows. In Section 2.1 we discuss exponential families in feature space and in Sections 2.2 -2.4 we present methods to deal with missing data. In Section 2.5 we show how Gaussian Processes and Support Vector Machines can be extended to deal with missing data. Section 3 is devoted to the discussion of the Constrained Concave Convex Procedure (CCCP) and its application to Gaussian Processes and Support Vector Machines. We discuss some implementation tips in Section 4 and present experimental results on the USPS dataset in Section 5. An outlook and a discussion in Section 6 conclude the paper.

2 The Model for Incomplete Data

2.1 Exponential Families

We begin with a definition of exponential families: Denote by $\mathcal{X}$ the domain of observations, and let $\phi(x)$ with $x \in \mathcal{X}$ refer to a vector of sufficient statistics.

Then, a member of the exponential family of densities can be defined in exponential normal form via

$$p(x;\theta) = p_0(x)\exp\left(\langle\phi(x),\theta\rangle - g(\theta)\right), \qquad (1)$$

where

$$g(\theta) = \log\int_{\mathcal{X}} p_0(x)\exp(\langle\phi(x),\theta\rangle)\,dx. \qquad (2)$$

Here, $p_0(x)$ is a suitably chosen underlying measure, θ is the natural parameter, $g(\theta)$ is the log-partition function, often called the cumulant generating function, and $\langle\cdot,\cdot\rangle$ denotes a scalar product in an Euclidean space, or more generally in a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$. Without loss of generality, and for ease of exposition, we will ignore the underlying measure $p_0(x)$ for the rest of the paper.

Let $\mathcal{Y}$ denote the space of labels, and $\phi(x,y)$ be the sufficient statistics of the joint distribution associated with $(x,y) \in \mathcal{X}\times\mathcal{Y}$. For the purpose of classification we are mainly concerned with estimating conditional probabilities. Therefore, we assume that given the data, the labels are drawn from an exponential family and, extend the exponential families framework to conditional probabilities. Here we have

$$p(y|x;\theta) = \exp\left(\langle\phi(x,y),\theta\rangle - g(\theta|x)\right), \qquad (3)$$

and

$$g(\theta|x) := \log\int_{\mathcal{Y}} \exp(\langle\phi(x,y),\theta\rangle)\,dy. \qquad (4)$$

In analogy to the above case, $g(\theta|x)$ is commonly referred to as the conditional log-partition function.

Both $g(\theta)$ and $g(\theta|x)$ are convex C^∞ functions in θ and they can be used to compute cumulants of the distribution [6, 4], for instance:

$$\begin{aligned}
\partial_\theta g(\theta) &= \mathbb{E}_{p(x;\theta)}[\phi(x)],\\
\partial^2_\theta g(\theta) &= \mathrm{Var}_{p(x;\theta)}[\phi(x)],\\
\partial_\theta g(\theta|x) &= \mathbb{E}_{p(x,y;\theta)}[\phi(x,y)|x],\\
\partial^2_\theta g(\theta|x) &= \mathrm{Var}_{p(x,y;\theta)}[\phi(x,y)|x].
\end{aligned}$$

2.2 Incomplete Training Data

In the following, we will deal with the problem of estimating $p(y|x;\theta)$ or a related quantity based on a set of observations $(x_i,y_i) \in \mathcal{X}\times\mathcal{Y}$ with $i = 1,\ldots,m$. More specifically, we allow that some of the x_i have been observed only partially, that is, we may partition the observations as $x_i = (x_i^o, x_i^u)$, where x_i^o represents the observed part and x_i^u is the unobserved part of the data (see [3] for a detailed description of how missing data may arise and how it is typically treated in an Expectation Maximization (EM) context). Observe that we allow for different sets of missing variables for different data points.

The first step is to extend (3) to partially observed data. Clearly

$$p(x^u,y|x^o;\theta) = \exp\left(\langle\phi(x^o,x^u,y),\theta\rangle - g(\theta|x^o)\right). \quad (5)$$

Integration over the unobserved part of x, that is, x^u, and direct calculation yields

$$\begin{aligned}
p(y|x^o;\theta) &= \int_{\mathcal{X}^u}\exp\left(\langle\phi(x^o,x^u,y),\theta\rangle - g(\theta|x^o)\right)dx^u\\
&= \exp(g(\theta|x^o,y) - g(\theta|x^o)), \qquad (6)
\end{aligned}$$

with a suitable definition of $g(\theta|x^o,y)$. In other words, the conditional probability $p(y|x^o;\theta)$ is now given by the exponential of the difference of two conditional log-partition functions. This poses two problems:

- Computing the log-partition function is a nontrivial problem [14]. In particular, the computation of $g(\theta|x^o)$ and $g(\theta|x^o,y)$ may pose additional difficulties. This is because the joint sufficient statistics might lead to an intractable integral. However, in many real life applications the data is discrete and only a small number of variables are missing. In these cases, one can either resort to brute force computation or exploit the algebraic structure of the integrand.
- The negative log-likelihood, $-\log(p(y|x^o;\theta))$, ceases to be a convex function. This means that the optimization problems arising from estimation with missing variables may involve many local optima. In Section 3, we will present an optimization method to deal with this problem by extending the CCP of [16] as well as a second method based on the EM algorithm.

2.3 Incomplete Labels

Using ideas similar to those used for handling missing training data we can also handle data with missing labels. As before, we partition $y_i = (y_i^o, y_i^u)$ and integrate out the unobserved part of the labels to yield

$$\begin{aligned}
p(y^o|x;\theta) &= \int_{\mathcal{Y}^u}\exp\left(\langle\phi(x,y^o,y^u),\theta\rangle - g(\theta|x)\right)dy^u\\
&= \exp(g(\theta|x,y^o) - g(\theta|x)). \qquad (7)
\end{aligned}$$

(7) can then be used to perform Maximum Likelihood Estimation (MLE) or Maximum A Posteriori (MAP) estimation. As before, the conditional probability $p(y^o|x;\theta)$ is given by the exponential of the difference of two conditional log-partition functions. Note that both types of missing data can also be combined in a straightforward manner using the conditional density $p(y^o|x^o;\theta)$.

2.4 Transduction

Transduction can be viewed as an extreme case of incomplete labels. Typically, we are given a set of observations with a few missing labels. The task is to predict these missing labels. We now compute a probability distribution on the missing labels, given by

$$p(y^u|x, y^o; \theta) = \exp\left(\langle \phi(x, y^o, y^u), \theta \rangle - g(\theta|x, y^o)\right).$$

In order to compute the above density we need to compute

$$g(\theta|x, y^o) = \log \int_{\mathcal{Y}^u} \exp(\langle \phi(x, y^o, y^u), \theta \rangle \; dy^u. \quad (8)$$

If the space of labels $\mathcal{Y}$ is large, and many labels are missing, then computing the above integral is a non-trivial task and we need to resort to Monte-Carlo sampling methods or other similar high dimensional integration techniques in order to perform prediction.

2.5 Conditional Probabilities and Estimators

Our discussion so far has been very generic. In this section, we focus on two particular kernel algorithms. First, we show how Gaussian Processes can be viewed as estimators in exponential families. Then, we discuss the well known Support Vector Machines in the context of exponential families. Using our discussion above, we also show how both these algorithms can handle missing data in a natural way.

Gaussian Process Classification: If the training data is assumed to be generated IID from an exponential family distribution, the MLE problem for exponential families is to minimize

$$\begin{aligned} -\log p(\theta|X, Y) &= \sum_{i=1}^{m} -\log p(y_i|x_i, \theta) \\ &= \sum_{i=1}^{m} g(\theta|x_i) - \langle \phi(x_i, y_i), \theta \rangle. \end{aligned}$$

Since we are considering an exponential family in feature space, the sufficient statistics are possibly infinite dimensional. To avoid over-fitting the data we consider a prior over the parameter θ.

One can show [1] that Gaussian Processes can be seen as estimators, where the prior on the natural parameter is normal, that is,

$$p(\theta) \propto \exp\left(-\frac{1}{2\sigma^2}\|\theta\|^2\right).$$

To see this, observe that under the above prior $t(x, y) := \langle \phi(x, y), \theta \rangle$ is a Gaussian Process. This is because θ is normally distributed with zero mean, and $\mathbb{E}_\theta[t(x, y)] = 0$ and the covariance (kernel) matrix is given by

$$k((x, y), (x', y')) = \langle \phi(x, y), \phi(x', y') \rangle.$$

This argument is similar to the one used by [15] to establish a connection between Support Vector Machines and Gaussian Processes.

As a special case we let $\mathcal{Y} = \{\pm 1\}$ and consider the choice $\phi(x, y) = y\phi'(x)$. This gives us $k((x, y), (x', y')) = y_i y_j \cdot k'(x, x')$ where $k'(x, x') = \langle \phi'(x), \phi'(x') \rangle$.

Now using the normal prior, the MAP estimation problem for exponential families is to minimize

$$\begin{aligned} -\log p(\theta|X, Y) &= \sum_{i=1}^{m} -\log p(y_i|x_i, \theta) + \frac{\|\theta\|^2}{2\sigma^2} \quad (9) \\ &= \sum_{i=1}^{m} g(\theta|x_i) - \langle \phi(x_i, y_i), \theta \rangle + \frac{\|\theta\|^2}{2\sigma^2}. \end{aligned}$$

Observe that the MAP estimation problem (9) is convex, and by the representer theorem [11], the minimizer θ^* can be found in the span of $\{\phi(x_i, y) \text{ where } y \in \mathcal{Y}\}$. So far, this interpretation of Gaussian Processes is consistent with the *classical* viewpoint.

We now turn to the setting with incomplete input data (the setting with missing labels is analogous and can be handled similarly). Here, all we need to do is to replace $p(y_i|x_i, \theta)$ by $p(y_i|x_i^o, \theta)$. Using (6) this leads to the following problem:

$$\text{minimize} \sum_{i=1}^{m} \left[g(\theta|x_i^o) - g(\theta|x_i^o, y_i)\right] + \frac{1}{2\sigma^2}\|\theta\|^2. \quad (10)$$

Unlike (9), the above problem is no longer convex, and we will need a more sophisticated method to solve it. In Section 3 we show how the CCCP method can be used to solve this optimization problem efficiently.

It is easy to check that $g(\theta|x_i^o, y_i) = \langle \phi(x_i^o, y_i), \theta \rangle$ if $x_i^o = x_i$, that is, we recover the original Gaussian Process optimization problem whenever the set of observations is complete.

Support Vector Classification: Gaussian Processes maximize the log-likelihood using a normal prior on the parameters. Instead of directly maximizing the log-likelihood, one may want to maximize the log-likelihood ratio between the correct label and the most likely incorrect labeling [9]. This leads to the following

cost function:

$$r(x, y; \theta) := \log \frac{p(y|x;\theta)}{\max_{\tilde{y} \neq y} p(\tilde{y}|x;\theta)} \tag{11}$$

$$= \langle \phi(x, y), \theta \rangle - \max_{\tilde{y} \neq y} \langle \phi(x, \tilde{y}), \theta \rangle . \tag{12}$$

In order to take the margin into account, we use

$$c(x, y; \theta) := \max(1 - r(x_i, y_i; \theta), 0)$$

which is essentially a clipped version of $r(x, y; \theta)$.

To see the connection to binary Support Vector Machines, assume $\mathcal{Y} \in \{\pm 1\}$ and $\phi(x, y) = \frac{y}{2}\phi'(x)$. Then, $r(x, y; \theta) = y_i \langle \phi'(x), \phi'(x') \rangle$ and $c(x, y; \theta) = \max(1 - y_i \langle \phi'(x), \phi(x') \rangle , 0)$ which essentially recovers the hinge loss. Therefore, our loss function is simply a generalization of the hinge loss to multi-class Support Vector Machines [9].

In fact, the MAP estimate in this case is found by solving

$$\operatorname*{argmin}_{\theta} \sum_{i=1}^{m} c(x_i, y_i; \theta) + \frac{1}{2\sigma^2}\|\theta\|^2. \tag{13}$$

To recover soft margin estimates, one simply needs to introduce slack variables into the above equation.

The main difference between the Support Vector Machine and the Gaussian process optimization problem is that, in the case of Support Vector Machines, the cost function $c(x, y; \theta)$ does *not* depend on the log-partition function. Instead, it is given by the difference between scalar products.

An extension to missing variables is now straightforward: all we need to do is to replace the conditional probability estimates in the fully observed case by their counterparts for partially observed data. Using (6) and (11) we have

$$r(x, y; \theta) = g(\theta|x^o, y) - \max_{\tilde{y} \neq y} g(\theta|x^o, \tilde{y}).$$

Finally, we can introduce slack variables and extend (13) into a constrained optimization problem for missing variables:

$$\text{minimize} \frac{1}{2\sigma^2}\|\theta\|^2 + \sum_{i=1}^{m} \xi_i \tag{14a}$$

$$\text{s.t. } g(\theta|x_i^o, y_i) - \max_{\tilde{y} \neq y_i} g(\theta|x_i^o, \tilde{y}) \geq 1 - \xi_i \tag{14b}$$

$$\xi_i \geq 0. \tag{14c}$$

The difference between (14) and (13) is that now the constraints, as specified by (14b), are non longer convex. Therefore, the minimization is no longer a convex problem, and we need, for instance, an iterative scheme to enforce these constraints.

As before, if $x_i^o = x_i$, that is, if no data is missing, we have $g(\theta|x_i^o, y_i) = \langle \phi(x_i^o, y_i), \theta \rangle$ and (14) reduces to a version of (13) which incorporates slack variables.

3 Optimization

As stated in Section 2.5, the optimization problems that arise when data is missing are no longer convex. Hence, it is a non-trivial task to solve them. In the case of Gaussian Processes one could invoke an EM like algorithm to perform maximum likelihood estimation over the joint set of parameters $(\theta, \{x_1^u, \ldots, x_m^u\})$ directly. But, it is not clear how such an algorithm can be extended to incorporate non-convex constraints which arise in the case of Support Vector Machines with missing variables.

Instead, we take a small detour: EM can also be viewed as a consequence of the CCP [16]. This provides us with a strategy to use similar algorithms for constrained problems by extending CCP to the Constrained CCP.

3.1 The Constrained Concave Convex Procedure

Theorem 1 (Constrained CCP) *Denote by f_i, g_j real-valued convex and differentiable functions on a vector space $\mathcal{X}$ for all $i \in \{0, \ldots, n\}$, and let $c_i \in \mathbb{R}$ for $i \in \{1, \ldots, n\}$. Then, Algorithm 1 converges to a local minimum of the following optimization problem, provided that the linearization of the nonconvex constraints in conjunction with the convex constraints satisfy suitable constraint qualifications at the point of convergence of the algorithm.*

$$\text{minimize} f_0(x) - g_0(x) \tag{15a}$$

$$\text{s.t. } f_i(x) - g_i(x) \leq c_i \text{ for all } 1 \leq i \leq n \tag{15b}$$

In the following, we denote by $T_n\{f, x\}(x')$ the n^{th} order Taylor expansion of f at location x, that is, $T_1\{f, x\}(x') = f(x) + \langle x' - x, \partial_x f(x) \rangle$.

Algorithm 1 Constrained Concave Convex Procedure

Initialize x_0 with a random value

repeat

find x_{t+1} as the solution of the convex optimization problem

$$\text{minimize} f_0(x) - T_1\{g_0, x_t\}(x) \tag{16a}$$

$$\text{s.t. } f_i(x) - T_1\{g_i, x_t\}(x) \leq c_i \, \forall i \tag{16b}$$

until convergence of x_t

Proof The key idea of the proof is that for any convex function, the first order Taylor expansion is a lower bound, that is, $g_i(x) \geq T_1\{g_i, x_t\}(x)$ for all $x, x_t \in \mathcal{X}$. Consequently for all $x, x_t \in \mathcal{X}$ and $0 \leq i \leq n$ we have

$$f_i(x) - T_1\{g_i, x_t\}(x) \geq f_i(x) - g_i(x). \tag{17}$$

By construction, equality holds at the point of expansion $x = x_t$. This means that for every x_t, (16) is an upper restriction of (15). In other words, every x feasible in (16b) is also feasible in (15b). Moreover, the objective function (16a) is an upper bound of (15a).

When $x = x_t$, the values of (15) and (16) match. Consequently, minimizing (16) leads to x_{t+1} with a lower value of the objective function (15a). This is because of two facts: Firstly, (16) presents an upper bound on (15). Secondly, when replacing the expansion at x_t by the one at x_{t+1} again the objective function may only decrease. To see this, observe that, by convexity we have

$$f_0(x_{t+1}) - T_1\{g_0, x_t\}(x_{t+1}) \geq f_0(x_{t+1}) - g_0(x_{t+1}),$$

but, by definition we have

$$f_0(x_{t+1}) - g_0(x_{t+1}) = f_0(x_{t+1}) - T_1\{g_0, x_{t+1}\}(x_{t+1}).$$

Next, we need to prove that if the x_t converge, then we actually arrived at a minimum or a saddlepoint of the optimization problem. We show this by proving that at stationarity a saddlepoint in the Lagrange function corresponding to (15) is also a saddlepoint in the Lagrange function corresponding to (16) with the same set of dual variables.

Now, assume that the above algorithm converges to x^* and let α^* be the dual variables of (16). By stationarity, the convex restriction at x^* satisfies the constraint qualifications and the Lagrange function of (16) has a saddle point in x^*, α^*.

However, by construction, the linearization is tight at x^*, so α^* also satisfies the Kuhn-Tucker conditions for (15a) and the derivatives of the Lagrangian of (15a) match those of their counterpart from (16a) at x^*. So we showed that if the convex restriction has a saddle point in the Lagrangian, so does the original problem. ■

This gives us a simple procedure to perform optimization even in a constrained nonconvex problem: simply linearize the constraints at every step and solve the resulting convex problem.

Remark 2 (CCP) *The CCP is a special case of theorem 1, where there are no constraints. In this case the first order conditions for the solution of (16a) amount to $\partial_\theta f_0(\theta) - \partial_\theta g_0(\theta_t) = 0$. This is exactly what [16] propose.*

3.2 Application to GP Classification

Recall that for Gaussian Process classification with missing variables the MAP-estimation problem (10) becomes that of solving

$$\text{minimize} \sum_{i=1}^{m} [g(\theta|x_i^o) - g(\theta|x_i^o, y_i)] + \frac{1}{2\sigma^2}\|\theta\|^2.$$

We define

$$\partial_\theta g(\theta|x^o) = \mathbb{E}_{p(x,y;\theta)}[\phi(x,y)|x^o;\theta] := E(\theta, x, y),$$

and

$$\partial_\theta g(\theta|x^o, y) = \mathbb{E}_{p(x,y;\theta)}[\phi(x,y)|x^o, y;\theta] := F(\theta, x, y).$$

Using the above, and the first-order optimality conditions of Remark 2, the Gaussian Process optimization problem can now be expressed as:

$$\sum_i E(\theta, x_i, y) - F(\theta_t, x_i, y_i) + \frac{1}{\sigma^2}\theta = 0. \tag{18}$$

Note that while the first expectation depends on θ, the second one is taken for a *fixed* value θ_t, which is the solution of the previous iteration of the optimization problem. We can now specialize Algorithm 1 to this case by iterating the above repeatedly with respect to θ. To show that our algorithm is identical to the EM algorithm, we show that an identical optimization problem arises out of the EM algorithm.

Recall that in the *expectation* step of EM one computes the value of the expected log-likelihood with respect to the given set of parameters θ_t, that is, we compute

$$\mathbf{E}_{p(x,y;\theta_t)}\left[\sum_{i=1}^{m} -\log p(y_i, x_i^u|x_i^o, \theta) + \frac{1}{2\sigma^2}\|\theta\|^2\right]. \tag{19}$$

Observe that

$$\mathbf{E}_{p(x,y;\theta_t)}\left[g(\theta|x_i^o)\right] = g(\theta|x_i^o),$$

and

$$\mathbf{E}_{p(x,y;\theta_t)}\left[\frac{1}{2\sigma^2}\|\theta\|^2\right] = \frac{1}{2\sigma^2}\|\theta\|^2.$$

Using the linearity of expectation, the above observations, and (5) we can re-write (19) as

$$\sum_{i=1}^{m} [g(\theta|x_i^o) - \langle F(\theta_t, x_i, y_i), \theta\rangle] + \frac{1}{2\sigma^2}\|\theta\|^2. \tag{20}$$

In the *maximization* step of EM, one computes the value of θ which maximizes the above expectation. First order optimality conditions for (20) are found by

taking derivatives with respect to θ and setting them to 0. This is equivalent to solving

$$\sum_i E(\theta, x_i, y) - F(\theta_t, x_i, y_i) + \frac{1}{\sigma^2}\theta = 0,$$

which is exactly the same as (18). This is not surprising, since the CCP is a generalization of the EM algorithm [16]. Things are more interesting in the case of Support Vector Machine classification.

3.3 Application to SV Classification

It is clear that (14) satisfies the conditions of Theorem 1: simply define

$$f_0(\theta, \xi) = \frac{1}{2\sigma^2}\|\theta\|^2 + \sum_{i=1}^m \xi_i \tag{21a}$$

$$f_i(\theta, \xi) = 1 - \xi_i + \max_{\tilde{y} \neq y_i} g(\theta | x_i^o, \tilde{y}) \tag{21b}$$

$$g_0(\theta) = 0 \text{ and } g_i(\theta) = g(\theta | x_i^o, y_i). \tag{21c}$$

We also set $c_i = 0$ for all i and write

$$T_1\{g_i, \theta_t\}(\theta) = g_i(\theta_t) + \langle \theta - \theta_t, F(\theta_t, x_i, y_i) \rangle .$$

If we define

$$d_i := 1 - \xi_i - g_i(\theta_t) + \langle \theta_t, F(\theta_t, x_i, y_i) \rangle ,$$

then each iteration Algorithm 1 requires solving the following optimization problem:

$$\min \ \frac{1}{2\sigma^2}\|\theta\|^2 + \sum_{i=1}^m \xi_i \tag{22a}$$

$$\text{s.t. } \langle F(\theta_t, x_i, y_i), \theta \rangle - \max_{\tilde{y} \neq y_i} g(\theta | x_i^o, \tilde{y}) \geq d_i \tag{22b}$$

$$\xi_i \geq 0. \tag{22c}$$

Since this is a convex optimization problem, standard Quadratic Programming (QP) packages can be used to solve it. Basically, what happens is that the expected value of $\Phi(x, y)$ with respect to the unknown part of x is used for classification. This is theoretical justification for the sometimes-used heuristic of estimating the values of the missing parameters and subsequently performing classification based on them. The main difference to this simple heuristic is that the margin of classification is defined as the difference between *pairs* of log-partition functions. This means that the conditional expectations depend on the (x^u, y) pair rather than on x^u alone.

4 Implementation

To make the above algorithms feasible in practice, several technical problems need to be overcome: it may not be possible to compute the log-partition function or its derivatives exactly. The solutions cease to be sparse, as they are given by linear combinations of conditional expectations. Sometimes, the dimensionality of the space might be so large that high dimensional integration techniques may need to be employed. In this section we discuss a few ideas which can be used to overcome the above problems.

The Representer Theorem: It follows from the generalized representer theorem [11] that the solution θ^* of both Support Vector Machine and Gaussian Process classification satisfies

$$\theta^* \in \text{span}\left\{\Phi(x_i^o, x_i^u, y) \text{ where } x_i^u, y \text{ are free}\right\}. \tag{23}$$

This means that the cardinality of the basis for θ is typically very large, sometimes even infinite. This might happen, for instance, when either the input space $\mathcal{X}$ or the label space $\mathcal{Y}$ have large dimensionality. This is clearly not desirable and we need an alternative. This is given in the form of an incomplete Cholesky factorization of the kernel matrix, either by sparse greedy approximation [13] or by positive diagonal pivoting [2]. For practical purposes we used the latter based on the kernel matrix arising from complete data pairs. The advantage is that instead of conditional expectations of $\Phi(x, y)$, which could be infinite dimensional, we now only need to compute conditional expectations over kernel values, that is

$$\langle \mathbf{E}_{x^u}[\Phi(x, y)|x^o; \theta], \Phi(x', y') \rangle = \mathbf{E}_{x^u}[k((x, y), (x', y'))|x^o]. \tag{24}$$

Likewise, second derivatives with respect to θ are given by covariances over kernel values.

In other words, instead of allowing the solution to lie in a possibly infinite dimensional space we constraint it to lie in a subspace spanned by the fully observed variables. This can lead to significant computational advantages. Of course, the downside is that the solution that we obtain might be sub-optimal since we are enforcing our constraint satisfaction conditions on only a subspace.

The log-partition function: The second issue, and arguably a very thorny one, is that one needs to be able to compute the value of the log-partition function for both the conditional as well as the unconditional densities. If suppose the number of missing variables is very small, and furthermore, if they can take only a small number of discrete values, then brute force computation of the conditional log-partition function is feasible.

In all other cases, we need to resort to methods for numerical quadrature, such as those discussed in [8].

The key difference to before is that now we will not even be able to reach a local optimum exactly but only up to the level of precision provided by the numerical integration method. A simple approximation is to use a Monte-Carlo estimate over the domain of missing variables instead of an exact integral. In other words, to compute

$$\mathbf{E}_{p(x,y;\theta)}[k((x,y),(x',y'))|x^o],$$

we use the approximation

$$\frac{\sum_{x^u \in X^u} k((x,y),(x',y'))e^{\langle \Phi(x,y),\theta\rangle}}{\sum_{x^u \in X^u} e^{\langle \Phi(x,y),\theta\rangle}}$$

where x^u is drawn uniformly from the domain of observations. We believe that the use of a more sophisticated MCMC sampling technique will lead to better performance.

Stochastic Gradient Descent Finally, instead of performing a new Taylor expansion at every new step, we may also perform stochastic gradient descent on the objective function itself. This may be preferable whenever the constrained optimization problem becomes highly nontrivial. Essentially, in this case we only perform conditional expectations for the particular observation at hand. Standard considerations for stochastic gradient descent methods apply [5].

5 Experiments

We use the well known US Postal Service (USPS) dataset. It contains 9298 handwritten digits (7291 for training and 2007 for testing), collected from mail envelopes in Buffalo [7]. Upto 25% of pixels (64 pixels out of 256) from each data point in the training set were randomly selected and their values were erased. A Sparse Greedy matrix approximation using a maximum of 1000 basis functions was used to approximate the kernel matrix. We use the Gaussian kernel

$$k(x,x') = \exp\left(\frac{\|x-x'\|^2}{2\sigma^2}\right),$$

and tune the σ parameter using cross validation. Regularization parameters previously reported in the literature [10] were used for all our experiments. To estimate the integrals we used a Monte-Carlo sampling technique using 50 configurations of missing data generated uniformly at random. We then used a block Jacobi method in conjunction with the CCCP algorithm in order to train a multi-class Gaussian Process. We obtained the best error rate of 5.8%. Contrast this with the best error rate of around 4.0% reported for the Gaussian kernel on the same dataset [10]. We noticed that estimating the integrals by using many samples decreases the error rate but takes significantly longer amounts of time to compute and converge.

In the second experiment, we replaced the missing values by their mean values from other observed data. This is commonly known as mean imputation [3]. We obtain the best error rate of 6.08% for this procedure.

As can be see the error rates achieved by our method is marginally better than that obtained by mean imputation. This phenomenon was also observed by [3]. We believe that more sophisticated numerical integration techniques to estimate integrals will significantly improve the performance of our algorithm.

6 Discussion and Outlook

In this paper, we presented a principled method for dealing with missing data using exponential families in feature space. We outlined methods to deal with missing training data as well as partially observed labels. Transduction can be viewed as a special case of our framework. We then showed how Gaussian Processes and Support Vector Machines can be extended to missing data by using our framework. In order to solve the non-convex optimization problem that arises we presented a generalization of the Convex Concave Procedure to incorporate non-convex constraints. We also discussed a simple proof of convergence for our algorithm. Preliminary experimental results are encouraging.

Clearly, computation of the log-partition function is the most expensive step in our algorithm. Faster approximation algorithms viz. Quasi Monte Carlo sampling methods need to be explored for computing the log-partition function. Extending our results to graphical models and other similar density estimators remains the focus of future research.

Acknowledgments National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council. This work was supported by grants of the ARC and sponsored by an NSF-ITR grant, award number IIS-0312401.

References

[1] Y. Altun, T. Hofmann, and A.J. Smola. Exponential families for conditional random fields. In *Uncertainty in Artificial Intelligence UAI*, 2004.

[2] S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243 – 264, Dec 2001. http://www.jmlr.org.

[3] Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via an EM approach. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 120 – 127. Morgan Kaufmann Publishers, Inc., 1994.

[4] R. E. Kass and P. W. Vos. *Geomtrical Foundations of Asymptotic Inference*. Wiley series in Probability and Statistics. Wiley Interscience, 1997.

[5] J. Kivinen, A.J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 2003. To Appear.

[6] S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[7] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. J. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541 – 551, 1989.

[8] W. H. Press, S. A. Teukolsky, W. T. Vetterhing, and B. P. Flannery. *Numerical Recipes in C. The Art of Scientific Computation*. Cambridge University Press, 1994.

[9] G. Rätsch, S. Mika, and A.J. Smola. Adapting codes and embeddings for polychotomies. In *Neural Information Processing Systems*, volume 15. MIT Press, 2002.

[10] B. Schölkopf. *Support Vector Learning*. R. Oldenbourg Verlag, München, 1997. Doktorarbeit, TU Berlin. Download: http://www.kernel-machines.org.

[11] B. Schölkopf, R. Herbrich, and A.J. Smola. A generalized representer theorem. In *Proceedings of the Annual Conference on Computational Learning Theory*, pages 416 – 426, 2001.

[12] B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, 2002.

[13] A.J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In P. Langley, editor, *Proceedings of the International Conference on Machine Learning*, pages 911 – 918, San Francisco, 2000. Morgan Kaufmann Publishers.

[14] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. Technical Report 649, UC Berkeley, Department of Statistics, September 2003.

[15] C. K. I. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning and Inference in Graphical Models*, pages 599 – 621. MIT Press, 1999.

[16] A.L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15:915 – 936, 2003.

Semiparametric Latent Factor Models

Yee Whye Teh
Computer Science Div.
University of California
Berkeley, CA 94720-1776
ywteh@eecs.berkeley.edu

Matthias Seeger
Computer Science Div.
University of California
Berkeley, CA 94720-1776
mseeger@eecs.berkeley.edu

Michael I. Jordan
Computer Science Div. and Dept. of Statistics
University of California
Berkeley, CA 94720-1776
jordan@eecs.berkeley.edu

Abstract

We propose a semiparametric model for regression problems involving multiple response variables. The model makes use of a set of Gaussian processes that are linearly mixed to capture dependencies that may exist among the response variables. We propose an efficient approximate inference scheme for this semiparametric model whose complexity is linear in the number of training data points. We present experimental results in the domain of multi-joint robot arm dynamics.

1 Introduction

We are interested in supervised problems involving multiple responses that we would like to model as conditionally dependent. In statistical terminology, we would like to "share statistical strength" between multiple response variables; in machine learning parlance this is often referred to as "transfer of learning." As we demonstrate empirically, such sharing can be especially powerful if the data for the responses is partially missing.

In this paper we focus on multivariate regression problems.[1] Models related to the one proposed here are used in geostatistics and spatial prediction under the name of *co-kriging* [3], and an example from this domain helps to give an idea of what we want to achieve with our technique. After an accidental uranium spill, a spatial map of uranium concentration is sought covering a limited area. We can take soil samples at locations of choice and measure their uranium content, and then use Gaussian process regression or another spatial prediction technique to infer a map. However, it is known that these carbon concentration and uranium concentration are often significantly correlated, and carbon concentration is easier to measure, allowing for more dense measurements. Thus in co-kriging the aim is to set up a joint spatial model for several responses with the aim of improving the prediction of one of them. The model to be described in the current paper goes beyond simple co-kriging methods in several ways. First, rather than combining responses in a posthoc manner as in co-kriging, our model uses latent random processes to represent *conditional* dependencies between responses directly. The latent processes are fitted using the data from all responses and can be used to model chacteristics of the dependencies beyond those based solely on marginal relationships. Second, the nature of the dependencies does not have to be known in advance but is learned from training data using empirical Bayesian techniques.

Another example of a motivating application arises in computer vision, where it is of interest to estimate the pose of a human figure from image data. In this case the response variables are the joint angles of the human body [1]. It is well known that human poses are highly constrained, and it would be useful for a pose estimation algorithm to take into account these strong dependencies among the joint angles.

Historically, the problem of capturing commonalities among multiple responses was one of the motivations behind multi-layer neural networks—the "hidden units" of a neural network were envisaged not only as nonlinear transformations, but also as adaptive basis functions to be "shared" in predicting multivariate responses. As neural networks gave way to kernel machines for classification and regression, with comcomitant improvements in flexibility, analytical tractability and performance, this core ability of neural networks was largely lost.

To elaborate on this point, note that there have been two main paths from neural networks to kernel machines. The first path, due to [10], involved the observation that in a particular limit the probability associated with (a Bayesian interpretation of) a neural network approaches a Gaussian process. For some purposes, it is arguably advantageous to work directly with the Gaussian process via its covariance function. However, in this limit it also turns out that the components of the response (the output

[1] In Section 5 we indicate how our technique can be extended to other settings such as multi-label classification.

vector) are independent—the ability to model couplings among these components is lost. The second path to kernel machines, via the optimization of margins [14], simplified the problem of fitting one-dimensional responses, but largely neglected the problem of fitting dependent multivariate responses. This problem has returned to the research agenda via architectures such as the conditional random field which links response variables using the graphical model formalism [5, 13].

Our approach to modeling dependencies among response variables heads in a direction that is more nonparametric than the CRF. In the spirit of factor analysis, we view the relationships among C components of a response vector $\boldsymbol{y}$ as reflecting a linear (or generalized linear) mixing of P underlying latent variables. These latent variables are indexed by a covariate vector $\boldsymbol{x}$, and thus we have a set of indexed collections of variables; that is, a set of stochastic processes. Specifically, we assume that each of the P variables is conditionally independently distributed according to a Gaussian process, with $\boldsymbol{x}$ as the (common) index set. The mean of the response $\boldsymbol{y}$ is then a (possibly nonlinear) function of a linear combination of these conditionally independent Gaussian processes.

This model is a semiparametric model, as it combines a nonparametric component (several Gaussian processes) with a parametric component (the linear mixing). We refer to the model as a *semiparametric latent factor model* (SLFM). Note that factor analysis is a special case of the SLFM, arising when $\boldsymbol{x}$ is a constant. Note also that Neal's limiting Gaussian process is a special case, arising when $P = 1$. Finally, as we discuss in Section 2, when $C = 1$ and $P > 1$ the SLFM can be viewed as a Gaussian process version of the multiple kernel learning architecture proposed in [6].

As in the case of simpler Gaussian process models, a significant part of the challenge of working with the SLFM is computational. This challenge can be largely met by exploiting recent developments in the literature on fitting large-scale Gaussian process regression and classification models. In particular, we make use of the informative vector machine (IVM) framework for Gaussian processes [8, 11]. In this framework, only a subset of "informative" likelihood terms are included in the computation of the posterior, yielding an training algorithm which scales linearly in the number of training data points. Moreover, the Bayesian underpinnings of the IVM yields general methods for setting free parameters (hyperparameters), an important capability which is not always easily achieved within the context of other kernel-based methodologies.

2 Semiparametric Latent Factor Models

In this section we give a description of our model for nonparametric regression with multiple responses. We begin with a short overview of Gaussian process (GP) regression in the simpler setting of single responses.

A GP can be viewed as a prior over random real-valued functions $u : \mathcal{X} \mapsto \mathbb{R}$, and is parametrized by a mean function $\mu(\cdot)$ and a covariance kernel $k(\cdot, \cdot)$. A random function $u(\cdot)$ is said to be distributed according to a GP if for any finite subset $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\} \subset \mathcal{X}$ of covariate vectors the random vector $u(X) = (u(\boldsymbol{x}_1), \ldots, u(\boldsymbol{x}_m))^T \in \mathbb{R}^m$ is distributed according to a Gaussian with mean $(\mu(\boldsymbol{x}_1), \ldots, \mu(\boldsymbol{x}_m))^T$ and covariance $\boldsymbol{K} \in \mathbb{R}^{m,m}$ where the i, j^{th} entry is $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. In our work we use GPs with zero mean: $\mu(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in \mathcal{X}$. The covariance kernel $k(\cdot, \cdot)$ has to satisfy a symmetric positive-definiteness (SPD) property; that is, $\boldsymbol{K}$ should be SPD for every finite subset X. Thus, a GP is simply a consistent way of assigning multivariate Gaussian distributions to $u(X)$ for any finite X.

GPs have traditionally been used for Bayesian classification and regression with a single response, where the problem is treated as that of estimating a random univariate function from the covariate space to the response space. Rather than assuming a parametric form for the random function, the nonparametric Bayesian approach places a GP prior over the space of all functions, and infers the posterior over functions given the training data.

Returning to our multiple response setting, let $\mathcal{X}$ be the covariate (input) space and let $\mathcal{Y} = \mathbb{R}^C$ be the response (output) space. We are interested in predicting $\boldsymbol{y} = (y_1, \ldots, y_C)^T \in \mathcal{Y}$ from $\boldsymbol{x} \in \mathcal{X}$; i.e., in estimating $P(\boldsymbol{y}|\boldsymbol{x})$. We model the conditional distribution using latent variables $\boldsymbol{v} \in \mathbb{R}^C$ such that the components y_c are mutually independent and independent of $\boldsymbol{x}$ given $\boldsymbol{v}$:

$$P(\boldsymbol{y}|\boldsymbol{v}, \boldsymbol{x}) = \prod_{c=1}^{C} P(y_c|v_c).$$

The conditional distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$ is then:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \int P(\boldsymbol{v}|\boldsymbol{x}) \prod_{c=1}^{C} P(y_c|v_c)\, d\boldsymbol{v}.$$

In this paper we focus on regression with Gaussian errors; i.e., $P(y_c|v_c) = N(y_c|v_c, \sigma_c^2)$. We treat $P(\boldsymbol{v}|\boldsymbol{x})$ nonparametrically, in particular using GPs. We do this by introducing a further set of latent variables $\boldsymbol{u} \in \mathbb{R}^P$ and letting $\boldsymbol{v}$ be linearly related to $\boldsymbol{u}$:

$$\boldsymbol{v} = \boldsymbol{\Phi}\boldsymbol{u}, \tag{1}$$

with $\boldsymbol{\Phi} \in \mathbb{R}^{C,P}$. Finally we assume that the coordinates of $\boldsymbol{u}$ have independent GP priors conditional on $\boldsymbol{x}$; i.e., there are random functions $u_p : \mathcal{X} \mapsto \mathbb{R}$ such that $u_p = u_p(\boldsymbol{x})$, and $u_p(\cdot)$ are distributed according to a GP with zero mean and covariance kernel $k_p(\cdot, \cdot)$. This setup allows

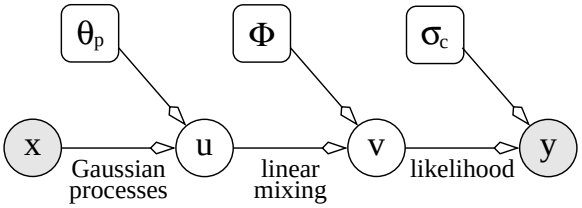


Figure 1: A semiparametric latent factor model.

for conditional dependencies among the coordinates of $\boldsymbol{v}$ to be expressed via $\boldsymbol{\Phi}$ and the latent $\boldsymbol{u}$ variables.

The form assumed in Eq. 1 is in direct analogy with factor analysis models. Note that the information learned from each single response variable y_c is reflected in the posterior for the latent GPs $\boldsymbol{u}(\cdot)$. Thus, there is sharing of statistical strength across response variables.

We call the model a *semiparametric latent factor model (SLFM)*; the graphical model is shown in Figure 1. The parameters are the components of $\boldsymbol{\Phi}$ and the hyperparameters (aka, the nuisance parameters) are the kernel parameters $\boldsymbol{\theta}_p$ and the variance parameters σ_c^2 associated with the Gaussian likelihoods $P(y_c|v_c)$.

Note that each coordinate $v_c(\cdot)$ of $\boldsymbol{v}(\cdot) = \boldsymbol{\Phi}\boldsymbol{u}(\cdot)$ is a priori a GP with covariance kernel given by $\sum_{p=1}^{P} \phi_{c,p}^2 k_p(\cdot,\cdot)$, where $\phi_{c,p}$ is the $(c,p)^{\text{th}}$ element of the matrix $\boldsymbol{\Phi}$. Hence one interpretation of our model is that each response variable is modeled as a Gaussian process with a kernel that is an adaptive, conic combination of base kernels. In fact, if we have $C = 1$ and $P > 1$, then this model can be viewed as a Gaussian process version of the kernel learning proposed in [6]. However our model does not just fit the kernel, it actually makes use of the same latent Gaussian processes for every response variable, allowing more expressive sharing of information across the response variates. Also, in the case $P < C$ which is our focus in the current paper, it is more efficient to represent $\boldsymbol{u}$ explicitly than to integrate it out.

3 Inference

Given a model and training samples $D = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$ drawn i.i.d. from the model[2], we perform Bayesian inference for the latent variables and estimate the parameters and the hyperparameters within an empirical Bayes framework. We begin by introducing the relevant latent variables.

Let $u_{i,p} = u_p(\boldsymbol{x}_i)$ and $v_{i,c} = v_c(\boldsymbol{x}_i)$. We collect these into vectors $\boldsymbol{u} = (u_{i,p})_{i,p}$ and $\boldsymbol{v} = (v_{i,c})_{i,c}$ where the double indices are flattened, with index i running over $1, \ldots, n$ first, e.g., $\boldsymbol{u} = (u_{1,1}, u_{2,1}, \ldots, u_{n,1}, \ldots, u_{n,P})^T$. The vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are again linearly related: $\boldsymbol{v} = (\boldsymbol{\Phi} \otimes \boldsymbol{I})\boldsymbol{u}$, where $\otimes$ is the Kronecker product. In the following we will assume that $P \quad C$ and $\boldsymbol{\Phi}$ has full rank so the pseudo-inverse $\boldsymbol{\Phi}^\dagger$ such that $\boldsymbol{\Phi}^\dagger\boldsymbol{\Phi} = \boldsymbol{I}$ exists and $\boldsymbol{u} = (\boldsymbol{\Phi}^\dagger \otimes \boldsymbol{I})\boldsymbol{v}$. The case $P > C$ requires a different treatment and will be presented in future work. The variable $\boldsymbol{u}$ is distributed a priori according to a Gaussian with zero mean and block diagonal covariance matrix $\boldsymbol{K} = \operatorname{diag}(\boldsymbol{K}^{(p)})_p$, where the p^{th} block has i, j^{th} entry given by $\boldsymbol{K}_{i,j}^{(p)} = k_p(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The covariance of $\boldsymbol{v}$ is thus $\tilde{\boldsymbol{K}} = (\boldsymbol{\Phi} \otimes \boldsymbol{I})\boldsymbol{K}(\boldsymbol{\Phi}^T \otimes \boldsymbol{I})$.

The posterior processes $\boldsymbol{u}(\cdot) \mid D$ are Gaussian in the case of Gaussian likelihoods $P(y_c|v_c)$, and in principle we could compute their mean and covariance functions explicitly. However, this is prohibitively expensive for all but fairly small values of n and C (the procedure scales as $O(n^3\, C^3)$ in general). We thus make use of the *informative vector machine (IVM)* framework [8] which computes a sparse approximation to the full Gaussian posterior $P(\boldsymbol{v}|D)$ by means of greedy forward selection of an *active subset* of the training sample using information-theoretic criteria. The difference here is that $P > 1$ processes have to be represented along with their dependencies, and the empirical Bayes maximization has to encompass a large number of non-kernel parameters, namely the elements of $\boldsymbol{\Phi}$.

3.1 Forward selection

The active set I of size d consists of tuples $(i,c) \in \{1, \ldots, n\} \times \{1, \ldots, C\}$. The goal is to select I such that the approximate posterior

$$Q(\boldsymbol{v}) \propto P(\boldsymbol{v}) \prod_{(i,c)\in I} P(y_{i,c}|v_{i,c}) \tag{2}$$

is close to $P(\boldsymbol{v}|D)$. For given I, the posterior approximation Q is given by simply ignoring all observations not in I. However, our method of selecting I depends on the complete sample D, as is discussed in this section. The idea is to greedily select the candidate (i,c) which changes the posterior most if we were to include it (i.e., incorporate its likelihood term into Q). A good way of measuring this change is the *information gain* studied in the setting of active learning (or sequential design). If Q_{k-1} denotes the posterior approximation after $k-1$ inclusions, the criterion is

$$\begin{aligned}\Delta_{i,c}^{info} &= \mathrm{D}\left[Q_{i,c}(\boldsymbol{v}) \,\|\, Q_{k-1}(\boldsymbol{v})\right] \\ &= \mathrm{D}\left[Q_{i,c}(v_{i,c}) \,\|\, Q_{k-1}(v_{i,c})\right],\end{aligned}$$

where $Q_{i,c}$ is the approximate posterior we obtain if the term (i,c) is included at iteration k. At each iteration we pick the (i,c) that maximizes $\Delta_{i,c}^{info}$. Since $Q_{i,c}(v_{i,c}) \propto P(y_{i,c}|v_{i,c})Q_{k-1}(v_{i,c})$, we can compute $\Delta_{i,c}^{info}$ in $O(1)$ if

[2] We allow *incomplete* observations of $\boldsymbol{y}_i$. That is, entries of $\boldsymbol{y}_i$ are allowed to be unobserved—this simply involves dropping likelihood terms corresponding to the unobserved entries.

the current marginal $Q_{k-1}(v_{i,c})$ is known. The representation of Q described in Section 3.2 makes it possible to maintain all these marginals at all times so that we can score all $(i,c) \notin I$ prior to each inclusion. After d iterations we have an active set $(i_1,c_1),\ldots,(i_k,c_k)$ which determines the approximate posterior in Eq. 2.

3.2 Representing the approximate posterior

The representation for the approximate posterior in Eq. 2 has to satisfy the following properties in order for it to be useful: it should allow all marginals $Q(v_{i,c})$ to be maintained explicitly at all times, which allows for forward selection (see Section 3.1); it should have a small memory footprint; and it can be efficiently updated when a new likelihood term is included. In this section we describe how these properties are achieved.

Let $\boldsymbol{\Pi} = \mathrm{diag}(\ _k)_k$ be a diagonal matrix and $\boldsymbol{b} = (b_k)_k$ be a vector which collects the parameters of the approximate posterior, in the sense that

$$Q(\boldsymbol{v}) \propto P(\boldsymbol{v}) \exp\ \ -\frac{1}{2}\boldsymbol{v}_I^T \boldsymbol{\Pi} \boldsymbol{v}_I + \boldsymbol{b}^T \boldsymbol{v}_I\ \ ,$$

where $\boldsymbol{v}_I = (v_{i_1,c_1},\ldots,v_{i_d,c_d})^T$. In the case of regression we have $_k = \sigma_{c_k}^{-2}$ and $b_k = \sigma_{c_k}^{-2} y_{i_k,c_k}$. Note that $\boldsymbol{\Pi}$ and $\boldsymbol{b}$ are ordered according to the order in which likelihood terms are included. To convert from this ordering to the natural ordering of $\boldsymbol{u}$ and $\boldsymbol{v}$, define a selection matrix $\boldsymbol{P} \in \mathbb{R}^{d,nC}$ where $\boldsymbol{P}_{k,(i_k,c_k)} = 1$ for $k = 1,\ldots,d$, and zeros elsewhere (note that the natural ordering in which we flatten i,c indices runs over i first).

Given $\boldsymbol{\Pi}$ and $\boldsymbol{P}$, the covariance of $Q_k(\boldsymbol{v})$ is

$$\tilde{\boldsymbol{A}} = (\tilde{\boldsymbol{K}}^\dagger + \boldsymbol{P}^T \boldsymbol{\Pi} \boldsymbol{P})^\dagger.$$

Given our assumption $P \quad C$, we can represent the approximate posterior in terms of $\boldsymbol{u}$ to minimize memory and time requirements. As $\boldsymbol{u} = (\boldsymbol{\Phi}^\dagger \otimes \boldsymbol{I})\boldsymbol{v}$, the covariance of $Q_k(\boldsymbol{u})$ is

$$\boldsymbol{A} = (\boldsymbol{\Phi}^\dagger \otimes \boldsymbol{I})(\tilde{\boldsymbol{K}}^\dagger + \boldsymbol{P}^T \boldsymbol{\Pi} \boldsymbol{P})^\dagger (\boldsymbol{\Phi}^{\dagger T} \otimes \boldsymbol{I}).$$

Using the Sherman-Morrison-Woodbury formula, we obtain

$$\boldsymbol{A} = \boldsymbol{K} - \boldsymbol{M}\boldsymbol{M}^T, \quad \boldsymbol{M} = \boldsymbol{K}(\boldsymbol{\Phi}^T \otimes \boldsymbol{I})\boldsymbol{P}^T \boldsymbol{\Pi}^{1/2} \boldsymbol{L}^{-T}, \tag{3}$$

where $\boldsymbol{L}$ is the lower triangular Cholesky factor of

$$\boldsymbol{B} = \boldsymbol{I} + \boldsymbol{\Pi}^{1/2} \boldsymbol{P} \tilde{\boldsymbol{K}} \boldsymbol{P}^T \boldsymbol{\Pi}^{1/2}.$$

The mean of $Q_k(\boldsymbol{u})$ is obtained as

$$\boldsymbol{h} = \mathrm{E}_{Q_k}[\boldsymbol{u}] = \boldsymbol{M}\boldsymbol{\beta}, \quad \boldsymbol{\beta} = \boldsymbol{L}^{-1} \boldsymbol{\Pi}^{-1/2} \boldsymbol{b} \tag{4}$$

while the mean and variance of $v_{i,c}$ under Q_k are

$$\tilde{h}_{i,c} = \boldsymbol{\phi}^{(c)} \boldsymbol{h}_i, \quad \tilde{a}_{i,c} = \boldsymbol{\phi}^{(c)} \boldsymbol{A}_i \boldsymbol{\phi}^{(c)T},$$

where $\boldsymbol{h}_i$ and $\boldsymbol{A}_i$ are the mean and covariance of $\boldsymbol{u}_i$, extracted from entries of Eq. 4 and Eq. 3 respectively, and $\boldsymbol{\phi}^{(c)}$ is the c^{th} row of $\boldsymbol{\Phi}$. The forward selection can be carried out once $\tilde{h}_{i,c}$ and $\tilde{a}_{i,c}$ are computed for every (i,c) not already included in the active set.

Finally, the representation of $Q(\boldsymbol{u})$ is given by $\boldsymbol{L} \in \mathbb{R}^{d,d}$, $\boldsymbol{\beta} \in \mathbb{R}^d$, $\boldsymbol{M} \in \mathbb{R}^{nP,d}$, and the mean $\boldsymbol{h}_i \in \mathbb{R}^P$ and covariance $\boldsymbol{A}_i \in \mathbb{R}^{P,P}$ of $\boldsymbol{u}_i$, for $i = 1,\ldots,n$. Since the rows of $\boldsymbol{L}$, $\boldsymbol{\beta}$, and $\boldsymbol{M}$ are already ordered by inclusion iteration, they can be updated efficiently and stably by appending new rows or columns. If at iteration k we included likelihood term (i,c), we compute

$$l = \sqrt{1 + \ _k \tilde{a}_{i,c}}, \qquad \boldsymbol{l} = \ _k^{1/2} \sum_p \phi_{c,p} \boldsymbol{M}^{(i,p)\ T}$$

$$\boldsymbol{\mu}_p = \frac{\phi_{c,p}\ _k^{1/2} \boldsymbol{K}_i^{(p)} - \boldsymbol{M}^{(p)} \boldsymbol{l}}{l} \qquad \gamma = \frac{\ _k^{-1/2} b_k - \boldsymbol{l}^T \boldsymbol{\beta}}{l},$$

where $\boldsymbol{M}^{(p)}$ is a matrix whose rows are $\boldsymbol{M}^{(i,p)}$. The new row of $\boldsymbol{L}$ is $(\boldsymbol{l}^T\ l)$, the new column of $\boldsymbol{M}^{(p)}$ is $\boldsymbol{\mu}_p$, and the new entry of $\boldsymbol{\beta}$ is γ. Let $\boldsymbol{\mu}_i = (\mu_{i,p})_p$. Then $\boldsymbol{h}_i$ is updated by adding $\gamma \boldsymbol{\mu}_i$ while $\boldsymbol{\mu}_i \boldsymbol{\mu}_i^T$ is subtracted from $\boldsymbol{A}_i$.

The memory requirement is dominated by $\boldsymbol{M}$ which is $O(nPd)$, while the P matrix-vector multiplications involved in computing $\boldsymbol{\mu}_p$ dominate the update time complexity, which is $O(nPd)$. Computing the information gain scores $\Delta_{i,c}^{info}$ requires $O(nCP^2)$ time which is in general subdominant to $O(nPd)$. Note that all costs are linear in the number of training points n which is the dominant factor in many large applications.

3.3 Parameter and hyperparameter estimation

We have described an effective procedure to compute an approximation to the posterior of the latent variables. Here we outline empirical Bayes estimation of the parameters and hyperparameters. Let $\boldsymbol{\alpha}$ denote a parameter or hyperparameter of interest, and let $\boldsymbol{s}$ denote the variational parameters which define the approximate posterior—these are the active set indicators, $\boldsymbol{\Pi}$ and $\boldsymbol{b}$. Since we cannot compute the marginal probability of $\boldsymbol{y}$ given $\boldsymbol{x}$ and $\boldsymbol{\alpha}$ exactly, we optimize a variational lower bound instead:

$$\log P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\alpha}) \geq \mathrm{E}_Q[\log P(\boldsymbol{y} \mid \boldsymbol{u}, \boldsymbol{\alpha})] - \mathrm{D}[Q(\boldsymbol{u}) \| P(\boldsymbol{u} \mid \boldsymbol{\alpha})], \tag{5}$$

where $Q(\boldsymbol{u})$ is the approximate posterior, given by Eq. 2. We use a double loop iterative procedure, where in the inner loop we optimize Eq. 5 with respect to $\boldsymbol{\alpha}$ using a quasi-Newton method while keeping $\boldsymbol{s}$ fixed, and in the outer

loop we reselect a new $\boldsymbol{s}$ greedily as detailed above. Notice that $Q(\cdot)$ is dependent on both $\boldsymbol{s}$ and $\boldsymbol{\alpha}$. For purposes of optimizing $\boldsymbol{\alpha}$ we propagate derivatives with respect to $\boldsymbol{\alpha}$ through $Q(\cdot)$, but keep $\boldsymbol{s}$ fixed. This differs from most other variational methods that keep all of $Q(\cdot)$ fixed when optimizing $\boldsymbol{\alpha}$. Note that the overall optimization is not guaranteed to converge, since the $\boldsymbol{s}$ updates are not guaranteed to increase the lower bound. In practice we find the optimization almost always increases and behaves well. The criterion and gradient computation has the same complexity as the conditional inference, but is much faster in practice because code for large matrix operations can be used. The memory requirements are not increased significantly, because $\boldsymbol{M}$ can be overwritten. The derivation is rather involved; it can be found in [12].

4 Experiments

In this section we present experimental results for the regression task of modeling of the dynamics of a 3-D, four-joint robot arm. The dataset is created using realistic simulation code which provides a mapping from twelve covariates (the angles, angular velocities and torques at each of the four joints of the arm) to four responses (the angular accelerations at the four joints).

We preprocessed the raw data by fitting a linear regression to the training set and replacing all responses by the corresponding residuals, then normalizing both covariate and response variables to have mean zero and variance one. This removal of the linear component of the regression helps clarify the relative contributions made by the nonlinear methods that are our focus. Finally the four responses were linearly mixed using randomly sampled unit length vectors to produce six response variables. Thus the dataset is a mapping from twelve covariates to six responses, where it is known that four latent variables are sufficient to capture the mapping.

The dataset sizes are $n = 1000$ for training and 500 points for testing. We report mean squared error (MSE) and average marginal log probability (LOGP) in the experiments below.[3] To calibrate the numbers, note that linear regression would have an average MSE of 1 on this task.

We compare our model against a baseline method (INDEP) in which each response variable is simply modeled independently, i.e., taking $P = C$ and $\boldsymbol{\Phi} = \boldsymbol{I}$. We use the IVM technique for both models. For our model we use a joint active set I of size d, while for the baseline we use individual active sets of size d' per response variate. It is clear that for similar training set size and coverage by active points, training for INDEP is significantly faster than for SLFM, and in this study we do not attempt to equalize training times.

[3] LOGP is $\log Q(y_*|\boldsymbol{x}_*)$ averaged over the test set.

In both models we use the *squared-exponential* covariance function (SQEXP):

$$k_p(\boldsymbol{x}, \boldsymbol{x}') = \ _p \exp \ \ -\sum_l \frac{1}{2\theta_{p,l}^2}|x_l - x_l'|^2 \Bigg), \quad (6)$$

where $\theta_{p,l}$ is the length scale for dimension l, and $\ _p > 0$ sets the overall variance. For the SLFM, the same kernel is used for all latent GPs and we set $\ _p = 1$ since the variance can already be represented by scaling the columns of $\boldsymbol{\Phi}$. We allow different length scales for each input dimension because we find that this has a significant impact on the quality of prediction—if the length scale is constrained to be the same for all covariate dimensions then less relevant input dimensions tend to obscure the more relevant ones. Note that this large set of hyperparameters is adjusted based on the training set within our empirical Bayesian framework; no validation set is needed.

The mean squared errors and log probabilities are shown in Table 1 as a function of P as P is varied from 2 to 6. The model performs best at $P = 4$, although similar accuracy is achieved for $P = 4, 5, 6$. We do expect the performance to degrade for even larger values of P but we have not investigated this. For the rest of this section we chose $P = 4$ since this is the smallest value supported by the data.

$c \setminus P$	2	3	4	5	6
1	0.2930	0.2340	0.1220	0.1190	0.1130
2	0.2840	0.2950	0.1780	0.1890	0.1880
3	0.3940	0.1570	0.1080	0.1060	0.1030
4	0.3630	0.2980	0.1270	0.1490	0.1410
5	0.3080	0.2830	0.1760	0.1840	0.1810
6	0.5560	0.3010	0.1180	0.1190	0.1100
LOGP	-5.770	-4.571	-2.342	-2.516	-2.466

Table 1: The mean squared errors for each response variate on the test set and the average log probability per training point assigned by our model for varying P.

Next we compared the SLFM with $P = 4$ to the baseline of independently modeled response variables. We used a training set of size $n = 1000$ and active sets of size $d = 1000$ and $d' = 180$ (if all INDEP active sets are disjoint, their union has size $6 \cdot 180 = 1080$). The results are shown in Table 2. Note that the MS errors for our model are smaller than for the baseline.

We also tested for the effects of varying the active set size d; results are given in Table 3.

We see that the active set size d has a significant effect on prediction accuracy. The quadratic scaling in d can be observed from the training times, except for the largest values of d, where the $O(d^3)$ component in the gradient computation dominates the $O(n\,P\,d^2)$ component.

Since our method models dependencies among the response variables, for every test point we have a joint predictive distribution over the response variables $\boldsymbol{y}_* \in \mathbb{R}^C$. This

c	SLFM MSE	INDEP MSE	SLFM LOGP	INDEP LOGP
1	0.122	0.133	0.018	-0.110
2	0.178	0.202	-0.244	-0.335
3	0.108	0.152	-0.025	-0.352
4	0.127	0.179	0.011	-0.271
5	0.176	0.202	-0.340	-0.349
6	0.118	0.135	0.053	-0.046

Table 2: Comparing our model (SLFM) against the baseline (INDEP) on the robot arm task, with $C = 6, P = 4$. Rows correspond to response variables.

$c \setminus d$	500	1000	2000	3000
1	0.174	0.122	0.096	0.067
2	0.285	0.178	0.107	0.094
3	0.228	0.108	0.072	0.062
4	0.283	0.127	0.082	0.068
5	0.281	0.176	0.100	0.090
6	0.196	0.118	0.090	0.066
time	382	1269	5806	16746

Table 3: MS test errors for each response as the active set size d is varied. *time* gives the complete training time in seconds.

can be used to further improve the prediction of the model for any specific component $y_{*,c}$, if in addition to the covariates $\boldsymbol{x}_*$ we are also given a subset of the other response variables $y_{*,c'}$. In Table 4 we show the mean squared errors attained for response $c = 5$, when we are also given other responses. The errors are reduced significantly, especially for $c' = \{2\}$, and further improve as we observe more responses; in particular when we observe $c' = \{3, 4\}$. Note that for the baseline method each response variable is modeled independently and the predictive distribution over $\boldsymbol{v}_*$ factorizes, hence knowledge of other responses cannot help in predicting $y_{*,c}$.

c'	MSE	LOGP
$\{1\}$	0.1770	-0.2640
$\{2\}$	0.0380	0.2450
$\{3\}$	0.1490	-0.2760
$\{4\}$	0.1320	-0.2940
$\{6\}$	0.1740	-0.3440
$\{3, 4\}$	0.112	-0.221

Table 4: Improved predictions of the model on response variable $c = 5$ when the model is given other responses $y_{*,c'}$.

Finally we report an experiment that aimed at improving our understanding of how statistical strength is shared across response variables. We again focus on predicting response variate $c = 5$ in the task with 1000 training points. However, instead of presenting all 1000 covariate/response pairs simultaneously, we start by observing only response variable $c = 5$, for a subset of $l < 1000$ points. Subsequently, we are given all 1000 covariate vectors and the corresponding responses for a subset c' not including $c = 5$. We ask whether this will improve our prediction of response variable $c = 5$. Note that this setup is similar to co-kriging scenarios mentioned in Section 1. Table 5 shows the mean squared errors attained for various values of l and for various subsets c' of additional observed response variables. We see a large improvement of the mean squared error for $c' = \{1, 2\}$. Even for $l = 50$ the errors are already smaller than those in Table 2. This is because of the strong dependencies between response variables $c = 2$ and $c = 5$ (as seen in Table 4). Note also that the case $c' = \{1, 2, 3, 4, 6\}$ performed worse than $c' = \{1, 2\}$, though still yielding a marked improvement over no additional training set ($c' = \emptyset$). This occurs because the functional that our method optimizes is a joint functional over the response variables, and thus depends more strongly on response variables with more training data. Performance as assessed by this functional indeed improved for larger c'. If our goal is to obtain good prediction for a particular response variable, we can consider a different functional which focuses on the response variable of interest.

$l \setminus c'$	$\emptyset$	$\{1, 2\}$	$\{1, 2, 3, 4, 6\}$
50	1.011	0.111	0.202
150	0.752	0.111	0.198
250	0.278	0.105	0.183

Table 5: Mean squared error on test set for varying training set sizes l and additional response variables c'.

5 Discussion

We have described a model for nonlinear regression in problems involving multiple, linked response variables. In a manner reminiscent of factor analysis in the parametric setting, we model the response vector as (a function of) a linear combination of a set of independent latent Gaussian processes. This rather simple semiparametric approach to sharing statistical strength has a number of virtues—most notably its flexible parametrization in terms of sets of covariance kernels and its computational tractability. We presented an efficient approximate inference strategy based on the IVM. While our primary focus has been prediction, the inferential tools provided by the IVM also allow us to compute posteriors over various components of the model, in particular the latent factors and the parameters. Possible extensions of the model include placing an automatic relevant determination (ARD) prior on the columns of the mixing matrix $\boldsymbol{\Phi}$ and letting the model determine P automatically. It is also of interest to consider ways in which the mixing matrix might be dependent on the covariates as well.

There are other ways of combining multiple Gaussian processes. [9] and [7] present models in which the hyperpa-

rameters of a set of Gaussian processes are endowed with a common prior. This hierarchical model couples the Gaussian processes as in the SLFM, but the amount of sharing that it induces is rather limited, since it involves only the hyperparameters of the Gaussian processes. In our approach the sharing involves entire processes and as such can be much more expressive. Note also that although we considered tasks involving a single regression problem with multiple responses, the SLFM can readily accommodate the setting in which there are multiple related tasks, each with a single response and with a separate training set.

As we have noted, the semiparametric approach presented here is an alternative to the parametric methodology of conditional random fields (CRFs) that has recently been the focus of attention in the machine learning and computer vision communities [5, 4]. When the response variables can plausibly be linked in a simple structure, for example according to a chain or a tree, the CRF approach would seem to be preferable to the SLFM approach. On the other hand, when the graph is not a chain, the potential intractability of the partition function can be a significant drawback of the CRF approach. In vision problems, for example, one would like to use a two-dimensional Markov random field for modeling dependencies, but this runs aground on the problem of the partition function. In our approach, couplings among variables arise by marginalizing over a latent set of linearly mixed Gaussian processes, and this provides an alternative, implicit approach to linking variables. In cases in which graphical models are intractable, this approach may provide the requisite tractability at a cost of modeling flexibility. Finally, note also that the SLFM approach is a kernel-based approach by definition; there is no need to explicitly "kernelize" the SLFM.

Several methods for multiple responses in regression have been proposed which involve posthoc combinations of the outputs of the independent baseline method. An example is the *curds and whey* method [2] for multiple linear regression. It is important to stress that our approach is fundamentally different in that the latent u processes are fitted jointly using all data. These processes can represent conditional dependencies directly, while the processes of the baseline method only ever see marginal data for each response. Posthoc combination schemes should be successful if response dependencies are mainly unconditional but may fail to represent dependencies which change with $\boldsymbol{x}$. An advantage of posthoc methods is that they can be cheap computationally, having essentially the same scaling as the independent baseline (which they use as a subroutine). Whether a more flexible technique such as ours with a computational complexity closer to the baseline method exists is an open question.

5.1 Applications to classification

Our model can be extended to classification problems and to other problems involving non-Gaussian likelihoods $P(y_c|v_c)$. The basic idea is to again make use of GP-based techniques such as the IVM that have been extended to GP-based classification in the single response variable case [8]. The non-Gaussian likelihoods are effectively replaced by Gaussians whose parameters are determined by sequential moment matching.

The extension to classification is of particular interest in the multiple response variable setting because it allows us to address multi-label classification problems in which the class labels are not assumed to be mutually exclusive and may exhibit interesting and useful interdependencies.

In a preliminary investigation of this extension we considered the toy example shown in Figure 2. We sampled 500 two-dimensional covariate vectors uniformly at random from $[-1, 1]^2$ and labeled these vectors using eight binary response variables, one for each of the regions shown in the top left panel of Figure 2. There was no label noise, but 10% of the $n = 500$ training responses were missing at random. We fit this data with an SLFM suitably extended to probit likelihoods, using $P = 3$ latent GPs, each with a different SQEXP kernel (Eq. 6) and with a single length scale parameter (i.e., $\theta_{p,l} = \theta_{p,l'}$).

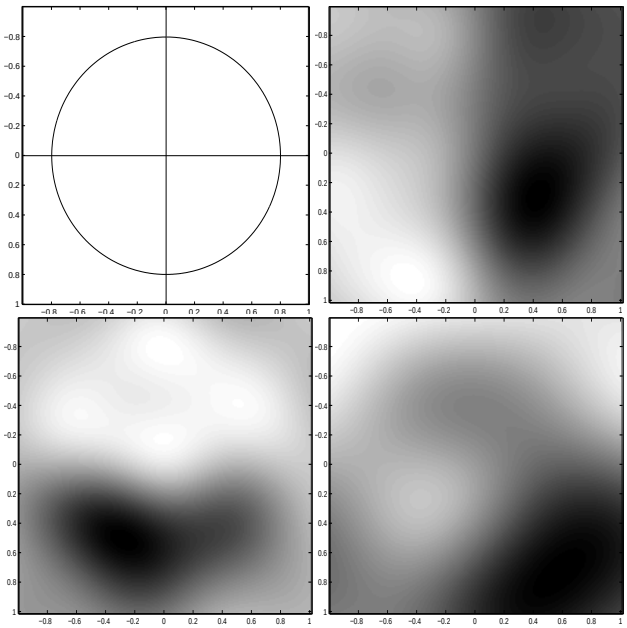


Figure 2: Top left: the eight regions. Rest: posterior mean function of the latent GPs. Light colors correspond to larger values.

After training, the test set errors for the eight response variables were 0.0345, 0.0261, 0.0133, 0.0296, 0.0602, 0.0176, 0.0494 and 0.0230. The remaining three panels in Figure 2 show the approximate posterior mean functions

for the three latent GPs. Roughly speaking, one GP is used for vertical discrimination, one for horizontal, and a third for inside-vs-outside separation (although the first two GPs also distinguish inside-vs-outside separation to a lesser extent). Thus we see that the model has formed a combinatorial code in which it is able to classify eight response variables using only three latent GPs.

5.2 Other issues

Computational issues remain a serious concern if the SFLM is to be scaled to larger problems. The main avenue open for tackling larger datasets is to refrain from scoring all remaining points for all later inclusions. In particular, after a number of initial inclusions selected from all remaining points we can potentially narrow down the candidate set stepwise by excluding points with the worst current scores. The empirical Bayes learning procedure then approximates the full likelihood by the likelihood restricted to the final candidate set (which always includes the active set). For very large tasks, even more elaborate caching strategies could be envisaged. A natural limit for the active set size d is imposed by the $O(d^3)$ time and memory scaling.

While we have focused on the case $P < C$ in the current paper, it is also of great interest to explore cases in which $P > C$. In particular, in the $P < C$ regime the variable $\boldsymbol{v} \in \mathbb{R}^C$ is constrained to lie in a P-dimensional subspace; while the analogy to factor analysis suggests that this may be a useful constraint in some problems, it also may impose an overly narrow bottleneck on the regression mapping in other problems. There are several possible ways to remove this constraint and consider versions of the SFLM that operate in the $P > C$ regime. One interesting variant involves replacing Eq. 1 by $\boldsymbol{v} = \boldsymbol{v}^{(0)} + \boldsymbol{\Phi}\boldsymbol{u}$ where all components of $(\boldsymbol{v}^{(0)T}, \boldsymbol{u}^T)^T$ are conditionally independent and are given GP priors with different kernels. While this setup can be viewed as simply a particular choice of $\boldsymbol{\Phi}$ in a generic SFLM with $P > C$, the additional independences in the model aid in the design of approximate inference methods based on a variant of belief propagation.

Acknowledgments

This work has been supported by a grant from the Intel Corporation, by a grant from Microsoft Research, and by a grant from DARPA in support of the CALO program.

References

[1] A. Agarwal and B. Triggs. 3D human pose from silhouettes by relevance vector regression. In *Proceedings of the IEEE International Conference on Computer vision and Pattern Recognition*, 2004.

[2] L. Breiman and J. Friedman. Predicting multivariate responses in multiple linear regression. *J. Roy. Stat. Soc. B*, 59(1):3–54, 1997.

[3] N. Cressie. *Statistics for Spatial Data*. John Wiley & Sons, 2nd edition, 1993.

[4] S. Kumar and M. Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *Proceedings of IEEE International Conference on Computer Vision*, 2003.

[5] J. Lafferty, F. Pereira, and A. McCallum. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, 2001.

[6] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[7] N. D. Lawrence and J. C. Platt. Learning to learn with the informative vector machine. In *Proceedings of the International Conference in Machine Learning*, 2004.

[8] N. D. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems 15*, pages 609–616, 2003.

[9] U. Menzefricke. Hierarchical modeling with Gaussian processes. *Communications in Statistics–Simulation and Computation*, 29(4):1089–1108, 2000.

[10] R. M. Neal. Priors for infinite networks. Technical Report CRG-TR-94-1, Department of Computer Science, University of Toronto, 1994.

[11] M. Seeger. *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, University of Edinburgh, July 2003. See `www.cs.berkeley.edu/~mseeger`.

[12] M. Seeger, Y.-W. Teh, and M. I. Jordan. Semiparametric latent factor models. Technical report, University of California at Berkeley, 2004. See `www.cs.berkeley.edu/~mseeger`.

[13] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16*, 2004.

[14] V. N. Vapnik. *Estimation of Dependences based on Empirical Data*. Series in Statistics. Springer, 1st edition, 1982.

Efficient Gradient Computation for Conditional Gaussian Models

Bo Thiesson
Microsoft Research
thiesson@microsoft.com

Christopher Meek
Microsoft Research
meek@microsoft.com

Abstract

We introduce Recursive Exponential Mixed Models (REMMs) and derive the gradient of the parameters for the incomplete-data likelihood. We demonstrate how one can use probabilistic inference in Conditional Gaussian (CG) graphical models, a special case of REMMs, to compute the gradient for a CG model. We also demonstrate that this approach can yield simple and effective algorithms for computing the gradient for models with tied parameters and illustrate this approach on stochastic ARMA models.

1 Introduction

The computation of parameter gradients given incomplete data is an important step in learning the parameters of a statistical model with missing data. In particular, for gradient based optimization methods, such as the conjugate gradient method, the gradient is used to iteratively adapt the parameters of the model in order to improve the incomplete-data likelihood and, in this way, identify the MLE or local maxima of the incomplete-data likelihood.

In this paper, we derive parameter gradients for a broad class of graphical models called recursive exponential mixed models (REMMs). REMMs generalize the well-known conditional Gaussian (CG) directed graphical models introduced by Lauritzen and Wermuth (1989). While REMMs have modelling advantages over CG models (e.g., allowing discrete variables to have continuous parents) our primary motivation for introducing REMMs in this paper is as a tool to derive the parameter gradient for CG models.

CG models are an important class of graphical models. They generalize discrete and Gaussian Bayesian networks and, importantly, they have efficient exact probabilistic inference algorithms for computing conditional marginal probabilities (Lauritzen and Jensen 2001). We derive the expression for the parameter gradient in CG models and demonstrate that standard probabilistic inference techniques can be used to efficiently compute these gradients.

We also demonstrate that our approach can yield simple and effective algorithms for computing parameter gradients of graphical models with tied parameters. We illustrate our approach to handling tied parameters on stochastic ARMA models (Thiesson *et al.* 2004). The stochastic ARMA model is of particular interest because it is a simple and useful model for modeling time-series data.

Previous papers have derived parameter gradients for graphical models. For instance, Thiesson (1997) and Binder *et al.* (1997) derive the parameter gradient for general classes of graphical models. In addition, both Thiesson (1997) and Binder *et al.* (1997) demonstrate that, for graphical models with only discrete variables, one can compute the parameter gradient using exact probabilistic inference. Binder *et al.* (1997) also discuss computation of the parameter gradient for models that have continuous variables. For such models, they resort to stochastic simulation to compute the gradient; they do so even with conditional Gaussian models for which exact probabilistic inference algorithms exist. Our results extend this work by demonstrating precisely how one can use probabilistic inference to compute the gradient for CG graphical models.

2 Recursive exponential mixed models

We consider directed graphical models with both discrete and continuous variables. For a directed graphical model the structural relations between variables $X = (X_v)_{v \in V}$, are represented by a directed acyclic graph (DAG), where each node v represents a variable, X_v, and directed edges represent direct influence from variables represented by parent nodes, $X_{pa(v)}$. Markov properties with respect to the graph (Kiiveri, Speed, and Carlin 1984; Lauritzen *et al.* 1990) imply that any distribution, which is structurally defined by a such model, can be represented by (local) conditional

distributions, $p(X_v|X_{pa(v)})$.

Thiesson (1997), defines a class of directed graphical models called recursive exponential models (REMs) for which the focus is on discrete variables. We extend this definition to mixed models with both continuous and discrete variables. We call this class of models for *recursive exponential mixed models* (REMMs).

Both REM and REMM models assume *global variation independence* for the parameters in the models. That is,

$$p(X|\theta) = \prod_{v \in V} p(X_v|X_{pa(v)}, \theta_v), \quad (1)$$

where $\Theta = \times_{v \in V} \Theta_v$, and $\theta_v \in \Theta_v$ completely specifies the relationship between the variable X_v and its conditional set of variables $X_{pa(v)}$.

For mixed models, the conditioning set for distributions on the right-hand side of (1) may have both continuous variables, denoted $X^c_{pa(v)}$, and discrete variables, denoted $X^d_{pa(v)}$. That is, $X_{pa(v)} = (X^c_{pa(v)}, X^d_{pa(v)})$. When the conditioning set contains discrete variables, REMM models will in addition assume *partial local variation independence* between parameters in conditional distributions with different values for the discrete conditioning variables. Let Π^d_v denote the set of all configurations for discrete parents of v, and let $\pi^d_v \in \Pi^d_v$ denote a particular configuration. By partial local parameter independence, $\Theta_v = \times_{\pi^d_v \in \Pi^d_v} \Theta_{v|\pi^d_v}$, and $\theta_{v|\pi^d_v} \in \Theta_{v|\pi^d_v}$ completely defines the *local model* $p(X_v|X^c_{pa(v)}, \pi^d_v, \theta_{v|\pi^d_v})$. Notice that if the discrete set of parent variables is empty, then the local model $p(X_v|X^c_{pa(v)}, \pi^d_v, \theta_{v|\pi^d_v}) = p(X_v|X_{pa(v)}, \theta_v)$. Hence, REMM models with only continuous variables, will only require global parameter independence. Notice also that if all involved variables are discrete, then partial local parameter independence is the same as local parameter independence, as defined for the REM models.

Given global and partial local parameter independence the likelihood for a single observation factors into *local likelihoods* as follows

$$p(x|\theta) = \prod_{v \in V} p(x_v|x_{pa(v)}, \theta_{v|\pi^d_v}).$$

For a REMM, the local models have to be representable as regular exponential models. Hence, a local likelihood is represented as

$$\begin{aligned} & p(x_v|x_{pa(v)}, \theta_{v|\pi^d_v}) \\ & \quad = \quad b(x_v) \exp\left(\theta_{v|\pi^d_v} t(x_v)' - \phi(\theta_{v|\pi^d_v})\right), \end{aligned} \quad (2)$$

where b is the carrying density, t the canonical statistics, ϕ the normalization function, and $'$ denotes transpose. Notice that b, t, ϕ are specific to the distribution, where we condition on the discrete parents $x^d_{pa(v)} = \pi^d_v$.

As described above, a model is a REMM if it is defined in terms of local models represented as regular exponential models and the collection of local models satisfy global and partial local variation independence. Later, in Section 5, we will see that the assumptions of variation independence can easily be relaxed to allow parameters to be tied across local models.

2.1 Conditional Gaussian models

The REMMs are particularly designed to generalize the class of conditional Gaussian (CG) models introduced by Lauritzen and Wermuth (1989). The CG models are of particular interest because we can, as we demonstrate below, use the exact inference scheme of Lauritzen and Jensen (2001) to efficiently compute the parameter gradients for these models.

A conditional Gaussian directed graphical models is a graphical model in which (i) the graphical DAG structure has no discrete variable with a continuous parent variable, (ii) the local models for discrete variables are defined by conditional multinomial distributions that can be represented in the usual way via conditional probability tables, and (iii) the local models for continuous variables (given continuous and discrete parents) are defind by conditional Gaussian regressions – one for each configuration of values for discrete parents. In particular

$$\begin{aligned} & p(X_v|X^c_{pa(v)}, \pi^d_v, \theta_{v|\pi^d_v}) \\ & \qquad \sim \quad \mathcal{N}\left(c(\pi^d_v) + \beta(\pi^d_v) X^c_{pa(v)}, \sigma(\pi^d_v)\right). \end{aligned}$$

We have here emphazised that the intercept for the regression, c, the linear regression coefficients, β, and the variance σ all depend on the particular configuration for the discrete parents, π^d_v. To simplify the notation in what follows, we will drop this explicit dependence.

Later, in Sections 4.2 and 4.3, we will see that local conditional multinomial and non-degenerate (or positive) local conditional Gaussian distributions can be represented as exponential models. A CG model assuming global and partial local parameter independence is therefore a REMM.

3 The incomplete-data gradient

We consider samples of incomplete observation and assume that the observations are incomplete in an non-informative way (e.g., missing at random; Gelman *et al.* 1995). Let $\mathbf{y} = (y^1, y^2, \ldots, y^L)$ denote a sample of possibly incomplete observations which are mutually independent. Given the mutual independence, the likelihood factorizes as a product over likelihoods for each observation

$$p(\mathbf{y}|\theta) = \prod_{l=1}^{L} p(y^l|\theta).$$

The gradient for the sample log-likelihood can therefore be obtained by simply adding the individual gradients for each observation. That is,

$$\frac{\partial \log p(\mathbf{y}|\theta)}{\partial \theta_{v|\pi_v^d}} = \sum_{l=1}^{L} \frac{\partial \log p(y^l|\theta)}{\partial \theta_{v|\pi_v^d}}. \quad (3)$$

We will in the next section derive the gradient expression for a single observation, knowing that the gradient for a sample can be obtained by simply adding up gradients for each observation, as in (3).

4 Single observation gradient

Suppose for a given model that a complete observation x is only observed indirectly through the *incomplete* observation y. Denote by $\mathcal{X}(y)$ the set of possible completions that are obtainable by augmenting the incomplete observation y. The likelihood for the incomplete observation then becomes

$$\begin{aligned} p(y|\theta) &= \int_{x\in\mathcal{X}(y)} p(x|\theta)\mu(x) \\ &= \int_{x\in\mathcal{X}(y)} \prod_{v\in V} p(x_v|x_{pa(v)}, \theta_{v|\pi_v^d})\mu(x), \quad (4) \end{aligned}$$

where μ is a generalized measure, which for a CG model is an appropriate combination of the counting measure for discrete variables and the Lebesque measure for continuous variables.

The gradient for the log-likelihood can now be expressed as

$$\begin{aligned} \frac{\partial \log p(y|\theta)}{\partial \theta_{v|\pi_v^d}} &= \frac{1}{p(y|\theta)} \frac{\partial p(y|\theta)}{\partial \theta_{v|\pi_v^d}} \\ &= \frac{1}{p(y|\theta)} \int_{x\in\mathcal{X}(y)} \frac{\partial p(x|\theta)}{\partial \theta_{v|\pi_v^d}} \mu(x), \quad (5) \end{aligned}$$

where the last equality follows from (4) and by using Leibnitz's rule for interchanging the order of differentiation and integration.

Now, consider the local gradient for the complete observation x. The chain rule for differentiation implies

$$\begin{aligned} \frac{\partial p(x|\theta)}{\partial \theta_{v|\pi_v^d}} &= \frac{p(x|\theta)}{p(x_v|x_{pa(v)}, \theta_{v|\pi_v^d})} \frac{\partial p(x_v|x_{pa(v)}, \theta_{v|\pi_v^d})}{\partial \theta_{v|\pi_v^d}} \\ &= p(x|\theta) \frac{\partial \log p(x_v|x_{pa(v)}, \theta_{v|\pi_v^d})}{\partial \theta_{v|\pi_v^d}}. \quad (6) \end{aligned}$$

Thus, by the exponential representation in (2), the local gradient for a complete observation becomes

$$\frac{\partial p(x|\theta)}{\partial \theta_{v|\pi_v^d}} = p(x|\theta) I^{\pi_v^d}(x^d_{pa(v)}) \left(t(x_v) - \tau(\theta_{v|\pi_v^d})\right), \quad (7)$$

where

$$\tau(\theta_{v|\pi_v^d}) = \frac{\partial \phi(\theta_{v|\pi_v^d})}{\partial \theta_{v|\pi_v^d}}$$

and $I^{\pi_v^d}(x^d_{pa(v)})$ is the indicator function, which is one for $x^d_{pa(v)} = \pi_v^d$ and zero otherwise.

It is a well-known fact from exponential model theory that the derivative for the normalizing function equals the expected value of the canonical statistics (see, e.g., Schervish 1995). That is

$$\tau(\theta_{v|\pi_v^d}) = \mathbf{E}_{\theta_{v|\pi_v^d}}[t(X_v)].$$

We will later use this fact when deriving the gradient for specific distributions.

Now, by inserting (7) into (5) we get the following expression for the local gradient of the incomplete observation

$$\begin{aligned} \frac{\partial \log p(y|\theta)}{\partial \theta_{v|\pi_v^d}} &= \int_{x\in\mathcal{X}(y)} \frac{p(x|\theta)}{p(y|\theta)} I^{\pi_v^d}(x^d_{pa(v)}) \\ &\quad \times \left(t(x_v) - \tau(\theta_{v|\pi_v^d})\right) \mu(x). \end{aligned}$$

Finally, by applying the fact that

$$p(x|y,\theta) = \begin{cases} \frac{p(x|\theta)}{p(y|\theta)} & \text{for } x \in \mathcal{X}(y) \text{ and } p(y|\theta) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

we obtain the final expression for the local gradient

$$\begin{aligned} &\frac{\partial \log p(y|\theta)}{\partial \theta_{v|\pi_v^d}} \\ &\quad = \int p(x_{v\cup pa(v)}|y,\theta) I^{\pi_v^d}(x^d_{pa(v)}) \\ &\qquad \times \left(t(x_v) - \tau(\theta_{v|\pi_v^d})\right) \mu(x_{v\cup pa(v)}) \\ &\quad = \int p(x_v, x^c_{pa(v)}, \pi_v^d|y,\theta) \\ &\qquad \times \left(t(x_v) - \tau(\theta_{v|\pi_v^d})\right) \mu(x_v, x^c_{pa(v)}) \quad (8) \end{aligned}$$

where the last equation follows from integrating over all discrete parents and exploiting the fact that $I^{\pi_v^d}(x^d_{pa(v)})$ is one for $x^d_{pa(v)} = \pi_v^d$ and zero otherwise.

The incomplete-data log-likelihood gradient expression in (8) apply for any local exponential model. This generality makes the expression appear somewhat complicated. However, as we will see below, in Sections 4.2 and 4.3, the exponential model expression leads to simple expressions for the more specific local conditional multinomials and conditional Gaussians in the CG models.

4.1 Re-parameterization

The local gradient for a regular exponential model distribution is a step on the way in deriving the local gradients for the specific local distributions in CG models. For these specific distributions, we will, however,

consider specific standard (non-exponential) parameterizations, as we will see in the subsections below. To obtain the gradient with respect to a new parameterization ψ, we apply the chain rule and multiply the derivative in (7) by the Jacobian $\partial\theta_{v|\pi_v^d}/\partial\psi$. Doing so, we obtain

$$\begin{aligned}\frac{\partial p(x|\psi)}{\partial\psi} &= \frac{\partial p(x|\theta)}{\partial\theta_{v|\pi_v^d}}\frac{\partial\theta_{v|\pi_v^d}}{\partial\psi} \\ &= p(x|\theta)I^{\pi_v^d}(x^d_{pa(v)}) \\ &\quad\times\left(t(x_v)-\tau(\theta_{v|\pi_v^d})\right)\frac{\partial\theta_{v|\pi_v^d}}{\partial\psi}.\end{aligned}$$

Performing the same operations that lead from (7) to (8) is trivial, and we finally obtain the expresion for the local gradient of the incomplete-data log-likelihood with repect to the re-parameterization

$$\begin{aligned}&\frac{\partial\log p(y|\psi)}{\partial\psi} \\ &= \int p(x_v, x^c_{pa(v)}, \pi_v^d|y,\theta) \\ &\quad\times\left(t(x_v)-\tau(\theta_{v|\pi_v^d})\right)\frac{\partial\theta_{v|\pi_v^d}}{\partial\psi}\mu(x_v, x^c_{pa(v)}). \quad (9)\end{aligned}$$

4.2 Conditional multinomial local gradient

Let us now take a look at the local gradient for the two specific types of local distributions in a CG model. First consider a *conditional multinomial* distribution for $p(X_v|\pi_v^d)$. As demonstrated in Thiesson (1997), we can obtain an exponential model representation for this distribution as follows. Let s_0 denote a value of reference for the discrete variable X_v and let $s_+ = 1,\ldots,S$ be the remaining possible values for X_v. If we choose s_0 as any value for which $p(s_0|\pi_v^d) > 0$, we can represent the conditional multinomial distribution by an exponential model with probabilities of the form (2) by letting

$$\begin{aligned}\theta^{s_+} &= \log\left[p(s_+|\pi_v^d)/p(s_0|\pi_v^d)\right] \\ t^{s_+}(x_v) &= \begin{cases}1 & \text{for } x_v = s_+ \\ 0 & \text{otherwise}\end{cases} \\ \phi(\theta_{v|\pi_v^d}) &= \log\left(1+\sum_{s_+=1}^{S}\exp(\theta^{s_+})\right) \\ b(x_v) &= 1\end{aligned}$$

where $\theta_{v|\pi_v^d} = (\theta^1,\ldots,\theta^S)$ and $t(x_v) = (t^1(x_v),\ldots,t^S(x_v))$.

The expected value for the canonical statistics in the above exponential model representation is

$$\begin{aligned}\tau(\theta_{v|\pi_v^d}) &= \mathbf{E}_{\theta_{v|\pi_v^d}}[t(X_v)] \\ &= \sum_{x_v} t(x_v)p(x_v|x_{pa(v)},\theta_{v|\pi_v^d}) \\ &= (p^1,\ldots,p^S)\end{aligned}$$

where $p^{s_+} = p(X_v = s_+|\pi_v^d,\theta_{v|\pi_v^d})$.

We finally obtain the expression for the local gradient with respect to the exponential model parameterization by inserting the above expressions for $t(x_v)$ and $\tau(\theta_{v|\pi_v^d})$ into equation (8). The elements of this vector are

$$\begin{aligned}&\frac{\partial\log p(y|\theta)}{\partial\theta^{s_+}} \\ &= \int p(x_v,\pi_v^d|y,\theta)t^{s_+}(x_v)\mu(x_v) \\ &\quad - \int p(x_v,\pi_v^d|y,\theta)p^{s_+}\mu(x_v) \\ &= p(s_+,\pi_v^d|y,\theta_{v|\pi_v^d}) \\ &\quad - p(\pi_v^d|y,\theta_{v|\pi_v^d})p(s_+|\pi_v^d,\theta_{v|\pi_v^d}). \quad (10)\end{aligned}$$

We can now use the Lauritzen and Jensen (2001) propagation scheme for Bayesian networks with CG distributions to efficiently compute the quantities in (10). The propagation scheme enables us to efficiently compute posterior marginal distributions for any family $X_{v\cup pa(v)}$ given evidence y. This marginal distribution is represented as the product of a marginal distribution for discrete variables and a conditional Gaussian for continuous variables given the discrete variables. The posterior probabilities $p(s_+,\pi_v^d|y,\theta_{v|\pi_v^d})$ and $p(\pi_v^d|y,\theta_{v|\pi_v^d})$ can therefore easily be extracted from the discrete marginal distribution and, hence, a conditional multinomial local gradient can be efficiently computed.

The Lauritzen and Jensen (2001) propagation scheme utilizes the traditional parameterization for the conditional multinomial distribution. This representation is given by the conditional probabilities $(p^0,\cdots,p^S)$, where $p^0 = p(X_v = s_0|\pi_v^d,\theta_{v|\pi_v^d})$ and p^{s_+} is defined as above. Hence, after we update the parameters for the exponential model representation during the line-search in a gradient based optimization method, we will need to switch back into the traditional representation – using (2) – in order to use the propagation scheme to compute the next gradient.

Switching between representations has a minor computational cost. On the other hand, performing the gradient optimization for parameters in the exponential model representation has the benefit that this parameterization automatically enforces the constraints $p^s \geq 0$ and $\sum_s p^s = 1$, which is not the case for gradient optimization using the traditional parameters.

We consider next the alternative gradient for the traditional parameter representation. In order to derive this gradient, we first derive the Jacobian from the ex-

ponential model representation to the traditional probability parameterization

$$\frac{\partial \theta_{v|\pi_v^d}}{\partial(p^0,\cdots,p^S)} = \begin{bmatrix} -1/p^0 & 1/p^1 & 0 & \dots & 0 \\ & & \vdots & & \\ -1/p^0 & 0 & \dots & 0 & 1/p^S \end{bmatrix}.$$

By insertion into equation (9) we now obtain the local gradient with respect to the traditional representation. The s^{th} $(s = 0, \dots, S)$ element in this gradient is given by

$$\frac{\partial \log p(y|\theta)}{\partial p^s} = \frac{p(s, \pi_v^d|y,\theta)}{p(s|\pi_v^d, \theta_{v|\pi_v^d})} - p(\pi_v^d|y,\theta). \quad (11)$$

Notice that the expression for this gradient differs slightly from the gradient expression in Binder *et al.* (1997) [Equation (4)].

Binder *et al.* (1997) ensure that the constraint $\sum_s p^s = 1$ is satisfied by using a standard method in which one projects the gradient onto the surface defined by this constraint. This method can be used for an optimization method based on our gradient in (11) as well. Still, however, both methods will have to ensure the constraint that $p^s \geq 0$ by inspecting the probability parameterization during a gradient update (i.e., a line-search).

4.3 Conditional Gaussian local gradient

Next, let us consider a *conditional Gaussian* (CG) local regression model for the continuous variable X_v given the parents $X_{pa(v)} = (X^c_{pa(v)}, X^d_{pa(v)})$, where the conditioning parent set may contain continuous variables, $X^c_{pa(v)}$, as well as discrete variables, $X^d_{pa(v)}$. Recall that π_v^d denotes a particular configuration of values for discrete parents, and to ease notation, we will in this section use a (instead of π_v^c or $x^c_{pa(v)}$) to denote a particular configuration of values for continuous parents. The CG regression model defines a set of linear regressions on the continuous parent variables – a regression for each configuration of distrete parent variables. See Lauritzen and Wermuth (1989) for more details on CG models. Let us now consider a particular distribution for X_v, given the values a for continuous parents and the configuration of values for discrete parents π_v^d. The distribution is defined by the mean $\mu = c + \beta a'$ and variance σ, where c and β are respectively the intercept and the coefficients for the regression on continuous parents, and $'$ denotes transpose. Restricting attention to non-degenerate (or positive) Gaussians, where $\sigma > 0$, we can obtain an exponential model representation of the form (2) as follows

$$\begin{aligned} \theta_{v|\pi_v^d} = (\theta_1, \theta_2) &= \left(\frac{\mu}{\sigma}, -\frac{1}{2\sigma}\right) \\ &= \left(\frac{c+\beta a'}{\sigma}, -\frac{1}{2\sigma}\right) \\ t(x_v) &= \left(x_v, x_v^2\right) \\ \phi(\theta_{v|\pi_v^d}) &= -\frac{\theta_1^2}{4\theta_2} - \frac{1}{2}\log(-2\theta_2) \\ &= \frac{\mu^2}{2\sigma} + \frac{1}{2}\log\sigma \\ b(x_v) &= (2\pi)^{-1/2}. \end{aligned}$$

Notice that the restriction to positive Gaussians ($\sigma > 0$) ensures that the natural parameters for the exponential representation are defined.

The expected value of the canonical statistics is

$$\begin{aligned} \tau(\theta_{v|\pi_v^d}) &= \mathbf{E}_{\theta_{v|\pi_v^d}}[t(X_v)] \\ &= (\mu, \sigma + \mu^2) \\ &= \left(c + \beta a', \sigma + (c+\beta a')^2\right). \end{aligned}$$

Again, we can use the Lauritzen and Jensen (2001) propagation scheme to efficiently compute the gradient for the parameters of the exponential model. As with the conditional multinomial case, the inference scheme utilizes a parameterization for a conditional Gaussian, that is different from our exponential model. In the case of conditional multinomial models, we can easily switch between parameterizations. This allowed us to use a gradient method (e.g., a line-search) to update the exponential model parameters and then convert the resulting parameterization back into the propagation scheme parameters in order to compute the next gradient. However, in the case of conditional Gaussian models, the propagation scheme requires the parameters (c, β, σ), and these parameters cannot be obtained from the parameters of the exponential model representation. We therefore compute the gradient for these parameters directly.

By inserting the expressions for $t(X_v)$ and $\tau(\theta_{v|\pi_v^d})$ into the equation (9) and by using the Jacobian

$$\frac{\partial \theta_{v|\pi_v^d}}{\partial(c,\beta,\sigma)} = \begin{bmatrix} \frac{1}{\sigma} & \frac{a}{\sigma} & -\frac{c+\beta a'}{\sigma^2} \\ 0 & 0 & \frac{1}{2\sigma^2} \end{bmatrix}$$

we can derive the following local gradient with respect to the parameterization (c, β, σ). Let $\mu = c + \beta a'$, then

$$\begin{aligned} &\frac{\partial \log p(y|\theta)}{\partial(c,\beta,\sigma)} \\ &= \frac{\partial \log p(y|\theta)}{\partial \theta_{v|\pi_v^d}} \frac{\partial \theta_{v|\pi_v^d}}{\partial(c,\beta,\sigma)} \\ &= \int p(x_v, a, \pi_v^d|y,\theta) \\ &\quad \times \begin{bmatrix} \frac{x_v-\mu}{\sigma} \\ \frac{(x_v-\mu)a}{\sigma} \\ \frac{(x_v-\mu)^2-\sigma}{2\sigma} \end{bmatrix}' \mu(x_v, a) \qquad (12) \end{aligned}$$

$$= p(\pi_v^d|y,\theta)\int p(x_v,a|\pi_v^d,y,\theta) \times \begin{bmatrix} \frac{x_v-\mu}{\sigma} \\ \frac{(x_v-\mu)a}{\sigma} \\ \frac{(x_v-\mu)^2-\sigma}{2\sigma} \end{bmatrix}' \mu(x_v,a) \quad (13)$$

where $p(x_v,a|\pi_v^d,y,\theta)=0$ for values of discrete parents π_v^d not consistent with the incomplete observation y. The step from (12) to (13) follows by factoring $p(x_v,a,\pi_v^d|y,\theta)$ into $p(x_v,a|\pi_v^d,y,\theta)$ and $p(\pi_v^d|y,\theta)$ and then pulling the discrete density out from under the integration.

Let $\overline{(x_v,a)}$ denote the expected value for the vector $(X_v,X^c_{pa(v)})$ with respect to the posterior Gaussian distribution for $(X_v,X^c_{pa(v)})$ given π_v^d That is,

$$\begin{aligned}\overline{(x_v,a)} &= \mathbf{E}_{\theta_{v|\pi_v^d}}[X_v,X^c_{pa(v)}|y] \\ &= \int p(x_v,a|\pi_v^d,y,\theta)(x_v,a)\mu(x_v,a).\end{aligned}$$

Similarly, let $\overline{(x_v,a)'(x_v,a)}$ denote the expected value for the matrix $\left((X_v,X^c_{pa(v)})'(X_v,X^c_{pa(v)})\right)$. For instance, $\overline{x_v a}=\mathbf{E}_{\theta_{v|\pi_v^d}}[X_v X^c_{pa(v)}|y]$.

The expression for the local gradient in (13) then reduces to

$$\begin{aligned}\frac{\partial \log p(y|\theta)}{\partial c} &= p(\pi_v^d|y,\theta)\left(\overline{x_v}-\beta\overline{a}-c\right)/\sigma \\ \frac{\partial \log p(y|\theta)}{\partial \beta} &= p(\pi_v^d|y,\theta)\left(\overline{x_v a}-c\overline{a}-\beta\overline{a'a}\right)/\sigma \\ \frac{\partial \log p(y|\theta)}{\partial \sigma} &= p(\pi_v^d|y,\theta)\left(\overline{x_v x_v}-2c\overline{x_v}-2\beta\overline{x_v a'}\right. \\ &\quad \left.+\beta\overline{a'a}\beta'+2c\beta\overline{a'}+c^2-\sigma\right)/2\sigma^2.\end{aligned} \quad (14)$$

We can now use the Lauritzen and Jensen (2001) propagation scheme and this time efficiently compute the gradient for CG regression models. Recall that the propagation scheme allow us to efficiently compute posterior marginal distributions for any family $X_{v\cup pa(v)}$, where this distribution is represented as the product of the marginal distribution for discrete variables, $p(X^d_{pa(v)})$, and the conditional Gaussian $p(X_v,X^c_{pa(v)}|X^d_{pa(v)})$. Given a particular configuration for the discrete variables, π_v^d, the mean vector μ and covariance matrix Σ for this conditional Gaussian equals

$$\begin{aligned}\mu &= \overline{(x_v,a)} \\ \Sigma &= \overline{(x_v,a)'(x_v,a)}-\mu'\mu.\end{aligned}$$

The expected statistics on the right-hand side of (14) can therefore easily be extracted from the parameterization of the marginal distribution and hence, the gradient for a CG regression can be efficiently computed.

5 Parameter tying

Tying of parameters is an essential feature for some types of models, including, for example, models for stochastic temporal processes and pedigree analysis. We consider parameter tying that relaxes the global variation independence in (1) by assuming that the parameterization for the relationship between the variable X_v and its conditional variables $X_{pa(v)}$ is the same across a set of variables. Let $\tilde{v}\subseteq V$ denote such a set of variables and let $\tilde{V}$ denotes all of such sets. We will let $\theta_{\tilde{v}}$ denote the tied parameterization across all $v\in\tilde{v}$. In this case, the model factorizes as

$$p(X|\theta)=\prod_{v\in V}p(X_v|X_{pa(v)},\theta_{\tilde{v}}) \quad (15)$$

where $\Theta=\times_{\tilde{v}\in\tilde{V}}\Theta_{\tilde{v}}$. We call this type of tying for *global parameter tying*. Global parameter tying is, of course, only possible between conditional models that are similar. That is, for all X_v, where $v\in\tilde{v}$, the number of discrete and continuous conditioning parent variables must be the same and the set of possible state configurations for discrete parents must be the same. We will let $\pi^d_{\tilde{v}}$ denote a particular configuration of states for discrete parent variables. This configuration will be the same across all $v\in\tilde{v}$.

We are now seeking the incomplete-data log-likelihood with respect to the parameterization $\theta_{\tilde{v}|\pi^d_{\tilde{v}}}$. Similar to (6) and (7), we use the chain rule and exponential model representation to first compute the local gradient for a complete observation

$$\begin{aligned}\frac{\partial p(x|\theta_{\tilde{v}})}{\partial \theta_{\tilde{v}|\pi^d_{\tilde{v}}}} &= \sum_{v\in\tilde{v}}p(x|\theta)\frac{\partial \log p(x_v|x_{pa(v)},\theta_{\tilde{v}|\pi^d_{\tilde{v}}})}{\partial\theta_{\tilde{v}|\pi^d_{\tilde{v}}}} \\ &= \sum_{v\in\tilde{v}}p(x|\theta)I^{\pi^d_{\tilde{v}}}(x_{pa(v)})\left(t(x_v)-\tau(\theta_{\tilde{v}|\pi^d_{\tilde{v}}})\right).\end{aligned}$$

We then obtain the expression for the local gradient for the incomplete data log-likelihood by the same steps, which lead to (8) and (9). Hence,

$$\begin{aligned}&\frac{\partial \log p(y|\psi)}{\partial\psi} \\ &=\sum_{v\in\tilde{v}}\int p(x_v,x^c_{pa(v)},\pi_{\tilde{v}|\pi^d_{\tilde{v}}}|y,\theta)\left(t(x_v)-\tau(\theta_{\tilde{v}|\pi^d_{\tilde{v}}})\right) \\ &\qquad\times\frac{\partial\theta_{\tilde{v}|\pi^d_{\tilde{v}}}}{\partial\psi}\mu(x_v,x^c_{pa(v)}).\end{aligned} \quad (16)$$

Setting $\frac{\partial\theta_{\tilde{v}|\pi^d_{\tilde{v}}}}{\partial\psi}=1$ gives us the expression for the gradient with respect to the natural parameters in the exponential model representation.

Notice that the only difference between (9) and (16) is that the gradient in (16) adds the gradients computed at each $v\in\tilde{v}$. In other words, with global

parameter tying, the gradient for the incomplete-data log-likelihood can be computed by proceeding as if parameters were not tied and then add up the gradients which are related by tying. That is,

$$\frac{\partial \log p(y|\psi)}{\partial \psi} = \sum_{v \in \tilde{v}} \frac{\partial \log p(y|\psi_{v|\pi_v^d})}{\partial \psi_{v|\pi_v^d}} \quad (17)$$

where $\psi_{v|\pi_v^d}$ denotes the (artificial) non-tied parameterization for the local model, with $\psi_{v|\pi_v^d} = \psi$ for all $v \in \tilde{v}$.

For simplicity, we will only consider global parameter tying, as describe above. More sophisticated tying schemes are, of course, possible.

5.1 Stochastic ARMA models

The stochastic ARMA (σARMA) models of Thiesson *et al.* (2004) is an illustrative example of a stochastic temporal process, where tying of parameters plays an important role. σARMA models are closely related to the classic autoregressive moving average (ARMA) time-series models (see, e.g., Box, Jenkins, and Reinsel 1994 or Ansley 1979). As demonstrated in Thiesson *et al.* (2004), both the ARMA and σARMA models are naturally represented as graphical models with only continuous variables. The σARMA models differs from the ARMA models by replacing the deterministic component of an ARMA model with a Gaussian distribution having a small variance, as we will see below. This variation allow us to smooth the time series model in a controlled way.

A σARMA(p,q) time-series model is defined as follows. We denote a temporal sequence of continuous observation variables by $Y = (Y_1, Y_2, \ldots, Y_T)$. Time-series data is a sequence of values for these variables – some of which may be missing. The models associate a latent "white noise" variable with each observable variable. These latent variables are denoted $E = (E_1, E_2, \ldots, E_T)$.

The σARMA model is now defined by the conditional Gaussian distribution

$$Y_t|Y_{t-p}, \ldots, Y_{t-1}, E_{t-q}, \ldots, E_t \sim \mathcal{N}(\mu_t, \sigma) \quad (18)$$

where the functional expression for the mean μ_t and the variance σ are shared across the observation variables. The variance is fixed at a given (small) value to be specified by the user. The mean is related to the conditional variables as follows

$$\mu_t = c + \sum_{j=0}^{q} \beta_j E_{t-j} + \sum_{i=1}^{p} \alpha_i Y_{t-i} \quad (19)$$

where c is the intercept for the regression, $\sum_{i=1}^{p} \alpha_i Y_{t-i}$ is the autoregressive (AR) part, $\sum_{j=0}^{q} \beta_j E_{t-j}$ is the moving average (MA) part with β_0 fixed as 1, and $E_t \sim \mathcal{N}(0, \gamma)$ with E_t mutually independent for all t. The model therefore involves the free parameters c, $(\alpha_1, \ldots, \alpha_p)$, $(\beta_1, \ldots, \beta_q)$, and γ. These parameters are tied across time steps.

From the above description, one may realize that an ARMA model is the limit of a σARMA model as $\sigma \to 0$. Letting $\sigma \to 0$ will in effect replace the conditional Gaussian distribution in (18) by a deterministic relation, where Y_t equals the right-hand side of (19).

We are interested in computing the gradient for the *conditional log-likelihood model*, where we condition on the first $R = max(p, q)$ variables. Relations between variables for $t \leq R$ can therefore be ignored. The graphical representation for an σARMA(2,2) model is shown in Figure 1. It should be noted that if we artificially extend the time series back in time for R (unobserved) time steps, this model represents what is known in the literature as the *exact likelihood model*. There are alternative methods for dealing with the beginning of a time series (see, e.g., Box, Jenkins, and Reinsel 1994).

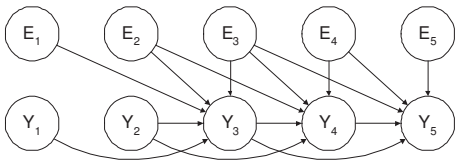


Figure 1: Graphical representation for σARMA(2,2) time-series model with five observations.

Let us first consider the variance parameter γ, which is tied across all the "white noise" variables $E_R, E_{R+1}, \ldots, E_T$. We intend to use (17) to compute the partial gradient for this parameter and will therefore first derive the expression for a partial gradient under the assumption that the γ parameters for these variables are not tied. Notice that γ will in this case take the place of the variance parameter σ in all of the formulas in Section 4.3. Also, recall that the "white noise" variables do not have any parents, which means that there are no regression coefficients and hence no partial derivative with respect to β. Because the Gaussian distribution is restricted to have a mean of zero, we invoke the chain-rule once more and multiply the gradient expression in (14) by the Jacobian $[0\ 1]'$ going from the (c, γ) parameterization to a parameterization, where γ is the only parameter. Notice that because the Jacobian is constant with respect to the integral in (13), the parameter gradient can be computed by simply multiplying the Jacobian and equation (14). As expected, we obtain a partial gradient for the non-tied variance of E_t that equals the partial derivative for the variance parameter in (14) – but it is not quite as complicated because $c = 0$ and E_t has no parents. Finally, by using (17), we arrive at the expression for the

partial gradient with respect to the tied γ parameter

$$\frac{\partial \log p(y|\theta)}{\partial \gamma} = \sum_{t=R+1}^{T} \frac{\overline{e_t e_t} - \gamma}{2\gamma^2}. \quad (20)$$

In a similar fashion, we can derive the gradient for the free parameters c, $(\alpha_1, \ldots, \alpha_p)$, and $(\beta_1, \ldots, \beta_q)$ associated with the conditional Gaussian distribution for the observation variables. As above, we apply the chain rule to achieve the gradient for the free parameters. Let $A_t = (Y_{t-p}, \ldots, Y_{t-1}, E_{t-q}, \ldots, E_t)$ denote all the parents for the observation variable Y_t and let $Z_t = A_t \setminus E_t$ denote the parents except for the parent E_t associated with the fixed regression coefficient β_0. We denote all of the regression coefficients by $\beta = (\alpha_1, \ldots, \alpha_p, \beta_0, \beta_1, \ldots, \beta_q)$ and let $\beta_{z_t} = (\alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_q)$ denote the free regression coefficients. The expression for the partial gradient for the tied c and β_{z_t} parameters now becomes

$$\frac{\partial \log p(y|\theta)}{\partial (c, \beta_{z_t})} = \sum_{t=R+1}^{T} \begin{bmatrix} (\overline{x_t} - \beta \overline{a_t} - c)/\sigma \\ \left(\overline{x_t z_t} - c\overline{z_t} - \beta_{z_t} \overline{z_t' z_t}\right)/\sigma \end{bmatrix}'.$$

6 Discussion and further work

In this paper, we derived the gradient for recursive exponential mixed models, a class of probabilistic models with both discrete and continuous variables. We demonstrated that positive conditional Gaussian models are a specific subclass of the REMMs and that one can use probabilistic inference to compute the parameter gradient for the incomplete-data likelihood for these models. As descried above, one can use this gradient to adapt the parameters in order to improve the incomplete-data likelihood and identify the MLE or local maxima of the likelihood. It is easy to extend this analysis to obtain similar results for MAP estimation by differentiating a prior with respect to the parameters of interest.

Alternative methods for learning parameters that do not directly compute the parameter gradient exist. For instance, the EM algorithm is a general method for improving parameters of a statistical model given incomplete data. In the context of graphical models, the E-step of the EM algorithm is accomplished via probabilistic inference in the graphical model (see, e.g., Lauritzen 1995 for a treatment of the EM algorithm for discrete graphical models). It should be noted, however, that in many situations, one can improve the speed of convergence of the EM algorithm through the use of the gradient (see, e.g., Thiesson 1995). Also, in some situations, the EM algorithm cannot be applied to improve the parameters of the model. In such situations a gradient method can often be used instead.

It is important to note that CG models that involve local degenerate conditional Gaussian models cannot be expressed as REMMs. The requirement for non-degenerate conditional Gaussians, where the variance $\sigma > 0$, can be seen by examining the exponential parameterization of the conditional Gaussian local model in Section 4.3. Unfortunately, some standard models can therefore not be naturally expressed as REMMs. For instance, the ARMA (a stochastic ARMA model in which the variance σ is zero) cannot be represented as a CG model. It is an open question as to whether or not probabilistic inference can be used to efficiently compute the gradient for non-positive CG models.

Finally, the class of REMMs has the advantage over CG models of allowing discrete variables to have continuous parents. Given this advantage, it would be worthwhile to investigate efficient methods for computing the parameter gradients for REMMs (i.e., the quantity in equation 8).

References

Ansley, C. F. (1979). An algorithm for the exact likelihood of a mixed autoregressive-moving average process. *Biometrika*, *66*, 59–65.

Binder, J., Koller, D., Russell, S. J., & Kanazawa, K. (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning*, *29*, 213–244.

Box, G. E. P., Jenkins, G. M., & Reinsel, G. C. (1994). *Time series analysis*. New Jersey: Prentice Hall.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (1995). *Bayesian data analysis*. London: Chapman and Hall.

Kiiveri, H., Speed, T. P., & Carlin, J. B. (1984). Recursive causal models. *Journal of the Australian Mathematical Society*, *36*, 30–52.

Lauritzen, S. L. (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics & Data Analysis*, *19*, 191–201.

Lauritzen, S. L., Dawid, A. P., Larsen, B. N., & Leimer, H.-G. (1990). Independence properties of directed Markov fields. *Networks*, *20*, 491–505.

Lauritzen, S. L., & Jensen, F. (2001). Stable local computation with conditional gaussian distributions. *Statistics and Computing*, *11*, 191–203.

Lauritzen, S. L., & Wermuth, N. (1989). Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, *17*, 31–57.

Schervish, M. J. (1995). *Theory of statistics*. New York: Springer-Verlag.

Thiesson, B. (1995). Accelerated quantification of Bayesian networks with incomplete data. *Proceedings of First International Conference on Knowledge Discovery and Data Mining* (pp. 306–311). AAAI Press.

Thiesson, B. (1997). Score and information for recursive exponential models with incomplete data. *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence* (pp. 453–463). Morgan Kaufmann Publishers.

Thiesson, B., Chickering, D. M., Heckerman, D., & Meek, C. (2004). ARMA time-series modeling with graphical models. *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence* (pp. 552–560). AUAI Press.

Very Large SVM Training using Core Vector Machines

Ivor W. Tsang **James T. Kwok** **Pak-Ming Cheung**
Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay
Hong Kong

Abstract

Standard SVM training has $O(m^3)$ time and $O(m^2)$ space complexities, where m is the training set size. In this paper, we scale up kernel methods by exploiting the "approximateness" in practical SVM implementations. We formulate many kernel methods as equivalent minimum enclosing ball problems in computational geometry, and then obtain provably approximately optimal solutions efficiently with the use of core-sets. Our proposed Core Vector Machine (CVM) algorithm has a time complexity that is *linear* in m and a space complexity that is *independent* of m. Experiments on large toy and real-world data sets demonstrate that the CVM is much faster and can handle much larger data sets than existing scale-up methods. In particular, on our PC with only 512M RAM, the CVM with Gaussian kernel can process the checkerboard data set with 1 million points in less than 13 seconds.

1 Introduction

In recent years, there has been a lot of interest on using kernels in various machine learning problems, with the support vector machines (SVM) being the most prominent example. Many of these kernel methods are formulated as quadratic programming (QP) problems. Denote the number of training patterns by m. The training time complexity of QP is $O(m^3)$ and its space complexity is at least quadratic. Hence, a major stumbling block is in scaling up these QP's to large data sets, such as those commonly encountered in data mining applications.

To reduce the time and space complexities, a popular technique is to obtain low-rank approximations on the kernel matrix, by using the Nyström method (Williams & Seeger, 2001), greedy approximation (Smola & Schölkopf, 2000) or matrix decompositions (Fine & Scheinberg, 2001). However, on very large data sets, the resulting rank of the kernel matrix may still be too high to be handled efficiently.

Another approach to scale up kernel methods is by chunking or more sophisticated decomposition methods. However, chunking needs to optimize the entire set of non-zero Lagrange multipliers that have been identified, and the resultant kernel matrix may still be too large to fit into memory. Osuna et al. (1997) suggested optimizing only a fixed-size subset of the training data (*working set*) each time, while the variables corresponding to the other patterns are frozen. Going to the extreme, the sequential minimal optimization (SMO) algorithm (Platt, 1999) breaks a large QP into a series of smallest possible QPs, each involving only two variables. In the context of classification, Mangasarian and Musicant (2001) proposed the Lagrangian SVM (LSVM) that avoids the QP (or LP) altogether. Instead, the solution is obtained by a fast iterative scheme. However, for nonlinear kernels (which is the focus in this paper), it still requires the inversion of an $m \times m$ matrix. Further speed-up is possible by employing the reduced SVM (RSVM) (Lee & Mangasarian, 2001), which uses a rectangular subset of the kernel matrix. However, this may lead to performance degradation (Lin & Lin, 2003).

In practice, state-of-the-art SVM implementations typically have a training time complexity that scales between $O(m)$ and $O(m^{2.3})$ (Platt, 1999). This can be further driven down to $O(m)$ with the use of a parallel mixture (Collobert et al., 2002). However, these are only empirical observations and not theoretical guarantees. For reliable scaling behavior to very large data sets, our goal is to develop an algorithm that can be proved (using tools in analysis of algorithms) to be asymptotically efficient in both time and space.

Moreover, practical SVM implementations, as in many numerical routines, only *approximate* the optimal solution by an iterative strategy. Typically, the stopping criterion utilizes either the precision of the Lagrange multipliers (e.g., (Joachims, 1999; Platt, 1999)) or the duality gap (e.g., (Smola & Schölkopf, 2004)). However, while approximation algorithms (with provable performance guarantees) have been extensively used in tackling computationally dif-

ficult problems like NP-complete problems (Garey & Johnson, 1979), such "approximateness" has never been exploited in the design of SVM implementations.

In this paper, we first transform the SVM optimization problem (with a possibly nonlinear kernel) to the *minimum enclosing ball* (MEB) problem in computational geometry. The MEB problem computes the ball of minimum radius enclosing a given set of points (or, more generally, balls). Traditional algorithms for finding exact MEBs do not scale well with the dimensionality d of the points. Consequently, recent attention has shifted to the development of approximation algorithms. Lately, a breakthrough was obtained by Bădoiu and Clarkson (2002), who showed that an $(1+\epsilon)$-approximation of the MEB can be efficiently obtained using *core-sets*. Generally speaking, in an optimization problem, a core-set is a subset of the input points, such that we can get a good approximation (with an approximation ratio[1] specified by a user-defined ϵ parameter) to the original input by solving the optimization problem directly on the core-set. Moreover, a surprising property of (Bădoiu & Clarkson, 2002) is that the size of its core-set is *independent* of both d and the size of the point set.

Inspired from this core-set-based approximate MEB algorithm, we will develop an approximation algorithm for SVM training that has an approximation ratio of $(1+\epsilon)^2$. Its time complexity is linear in m while its space complexity is independent of m. The rest of this paper is organized as follows. Section 2 gives a short introduction on the MEB problem and its approximation algorithm. The connection between kernel methods and the MEB problem is given in Section 3. Section 4 then describes our proposed Core Vector Machine (CVM) algorithm. Experimental results are presented in Section 5, and the last section gives some concluding remarks.

2 MEB in Computational Geometry

Given a set of points $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, where each $\mathbf{x}_i \in \mathbb{R}^d$, the minimum enclosing ball of $\mathcal{S}$ (denoted MEB($\mathcal{S}$)) is the smallest ball that contains all the points in $\mathcal{S}$. The MEB problem has found applications in diverse areas such as computer graphics (e.g., collision detection, visibility culling), machine learning (e.g., similarity search) and facility locations problems.

[1] Let C be the cost (or value of the objective function) of the solution returned by an approximate algorithm, and C^* be the cost of the optimal solution. Then, the approximate algorithm has an *approximation ratio* $\rho(n)$ for an input size n if $\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq \rho(n)$. Intuitively, this measures how bad the approximate solution is compared with the optimal solution. A large (small) approximation ratio means the solution is much worse than (more or less the same as) the optimal solution. Observe that $\rho(n)$ is always ≥ 1. If the ratio does not depend on n, we may just write ρ and call the algorithm an *ρ-approximation algorithm*.

Here, we will focus on approximate MEB algorithms based on core-sets. Let $B(\mathbf{c}, R)$ be the ball with center $\mathbf{c}$ and radius R. Given $\epsilon > 0$, a ball $B(\mathbf{c}, (1+\epsilon)R)$ is an *$(1+\epsilon)$-approximation* of MEB($\mathcal{S}$) if $R \leq r_{\text{MEB}(\mathcal{S})}$ and $\mathcal{S} \subset B(\mathbf{c}, (1+\epsilon)R)$. A subset $X \subseteq \mathcal{S}$ is a *core-set* of $\mathcal{S}$ if an expansion by a factor $(1+\epsilon)$ of its MEB contains $\mathcal{S}$, i.e., $\mathcal{S} \subset B(\mathbf{c}, (1+\epsilon)r)$, where $B(\mathbf{c}, r) = \text{MEB}(X)$ (Figure 1).

To obtain such an $(1+\epsilon)$-approximation, Bădoiu and Clarkson (2002) proposed a simple iterative scheme: At the tth iteration, the current estimate $B(\mathbf{c}_t, r_t)$ is expanded incrementally by including the furthest point outside the $(1+\epsilon)$-ball $B(\mathbf{c}_t, (1+\epsilon)r_t)$. This is repeated until all the points in $\mathcal{S}$ are covered by $B(\mathbf{c}_t, (1+\epsilon)r_t)$. Despite its simplicity, Bădoiu and Clarkson (2002) showed that the number of iterations, and hence the size of the final core-set, depends only on ϵ but *not* on d or m. This independence of d is important on applying this algorithm to kernel methods (Section 3) as the kernel-induced feature space can be infinite-dimensional. As for the independence on m, it allows both the time and space complexities of our algorithm to grow slowly, as will be shown in Section 4.3.

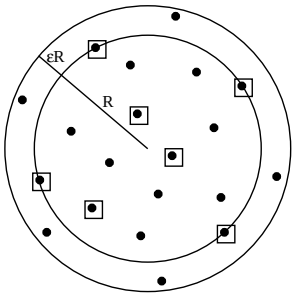


Figure 1: The inner circle is the MEB of the set of squares and its $(1+\epsilon)$ expansion (the outer circle) covers all the points. The set of squares is thus a core-set.

3 MEB Problems and Kernel Methods

Obviously, the MEB is equivalent to the hard-margin support vector data description (SVDD) (Tax & Duin, 1999), which will be briefly reviewed in Section 3.1. The MEB problem can also be used for finding the radius component of the radius-margin bound (Chapelle et al., 2002). Thus, as pointed out by Kumar et al. (2003), the MEB problem is useful in support vector clustering and SVM parameter tuning. However, we will show in Section 3.2 that other kernel-related problems, including the training of soft-margin one-class and two-class L2-SVMs, can also be viewed as MEB problems.

3.1 Hard-Margin SVDD

Given a kernel k with the associated feature map φ, let the MEB in the kernel-induced feature space be $B(\mathbf{c}, R)$. The primal problem in the hard-margin SVDD is

$$\min R^2 \;\; : \;\; \|\mathbf{c} - \varphi(\mathbf{x}_i)\|^2 \leq R^2, \;\; i = 1, \ldots, m. \quad (1)$$

The corresponding dual is

$$\max \boldsymbol{\alpha}' \text{diag}(\mathbf{K}) - \boldsymbol{\alpha}' \mathbf{K} \boldsymbol{\alpha} \;\; : \;\; \mathbf{0} \leq \boldsymbol{\alpha}, \;\; \boldsymbol{\alpha}' \mathbf{1} = 1, \quad (2)$$

where $\boldsymbol{\alpha} = [\alpha_i, \ldots, \alpha_m]'$ are the Lagrange multipliers, $\mathbf{0} = [0, \ldots, 0]'$, $\mathbf{1} = [1, \ldots, 1]'$ and $\mathbf{K}_{m\times m} = [k(\mathbf{x}_i, \mathbf{x}_j)] = [\varphi(\mathbf{x}_i)'\varphi(\mathbf{x}_j)]$ is the kernel matrix. As is well-known, this is a QP problem. The primal variables can be recovered from the optimal $\boldsymbol{\alpha}$ as

$$\mathbf{c} = \sum_{i=1}^{m} \alpha_i \varphi(\mathbf{x}_i), \quad R = \sqrt{\boldsymbol{\alpha}'\text{diag}(\mathbf{K}) - \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}}. \quad (3)$$

3.2 Viewing Kernel Methods as MEB Problems

In this paper, we consider the situation where

$$k(\mathbf{x}, \mathbf{x}) = \kappa, \quad (4)$$

a constant[2]. This will be the case when either (1) the isotropic kernel $k(\mathbf{x}, \mathbf{y}) = K(\|\mathbf{x}-\mathbf{y}\|)$ (e.g., Gaussian kernel); or (2) the dot product kernel $k(\mathbf{x}, \mathbf{y}) = K(\mathbf{x}'\mathbf{y})$ (e.g., polynomial kernel) with normalized inputs; or (3) any normalized kernel $k(\mathbf{x}, \mathbf{y}) = \frac{K(\mathbf{x},\mathbf{y})}{\sqrt{K(\mathbf{x},\mathbf{x})}\sqrt{K(\mathbf{y},\mathbf{y})}}$ is used. Using the condition $\boldsymbol{\alpha}'\mathbf{1} = 1$ in (2), we have $\boldsymbol{\alpha}'\text{diag}(\mathbf{K}) = \kappa$. Dropping this constant term from the dual objective in (2), we obtain a simpler optimization problem:

$$\max -\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \;:\; \mathbf{0} \le \boldsymbol{\alpha}, \;\; \boldsymbol{\alpha}'\mathbf{1} = 1. \quad (5)$$

Conversely, when the kernel k satisfies (4), QP's of the form (5) can always be regarded as a MEB problem (1). Note that (2) and (5) yield the same set of $\boldsymbol{\alpha}$'s, Moreover, let d_1^* and d_2^* denote the optimal dual objectives in (2) and (5) respectively, then, obviously,

$$d_1^* = d_2^* + \kappa. \quad (6)$$

In the following, we will show that when (4) is satisfied, the duals in a number of kernel methods can be rewritten in the form of (5). While the 1-norm error has been commonly used for the SVM, our main focus will be on the 2-norm error. In theory, this could be less robust in the presence of outliers. However, experimentally, its generalization performance is often comparable to that of the L1-SVM (e.g., (Lee & Mangasarian, 2001; Mangasarian & Musicant, 2001). Besides, the 2-norm error is more advantageous here because a soft-margin L2-SVM can be transformed to a hard-margin one. While the 2-norm error has been used in classification (Section 3.2.2), we will also extend its use for novelty detection (Section 3.2.1).

3.2.1 One-Class L2-SVM

Given a set of unlabeled patterns $\{\mathbf{z}_i\}_{i=1}^m$ where $\mathbf{z}_i$ only has the input part $\mathbf{x}_i$, the one-class L2-SVM separates the outliers from the normal data by solving the primal problem:

$$\min_{\mathbf{w},\rho,\xi_i} \|\mathbf{w}\|^2 - 2\rho + C\sum_{i=1}^{m} \xi_i^2 \;:\; \mathbf{w}'\varphi(\mathbf{x}_i) \ge \rho - \xi_i,$$

where $\mathbf{w}'\varphi(\mathbf{x}) = \rho$ is the desired hyperplane and C is a user-defined parameter. Note that constraints $\xi_i \ge 0$ are not needed for the L2-SVM. The corresponding dual is

$$\begin{aligned} & \max -\boldsymbol{\alpha}'\left(\mathbf{K} + \frac{1}{C}\mathbf{I}\right)\boldsymbol{\alpha} \;:\; \mathbf{0} \le \boldsymbol{\alpha}, \;\; \boldsymbol{\alpha}'\mathbf{1} = 1 \\ = \;& \max -\boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha} \;:\; \mathbf{0} \le \boldsymbol{\alpha}, \;\; \boldsymbol{\alpha}'\mathbf{1} = 1, \end{aligned} \quad (7)$$

where $\mathbf{I}$ is the $m\times m$ identity matrix and $\tilde{\mathbf{K}} = [\tilde{k}(\mathbf{z}_i, \mathbf{z}_j)] = [k(\mathbf{x}_i, \mathbf{x}_j) + \frac{\delta_{ij}}{C}]$. It is thus of the form in (5). Since $k(\mathbf{x}, \mathbf{x}) = \kappa$, $\tilde{k}(\mathbf{z}, \mathbf{z}) = \kappa + \frac{1}{C} \equiv \tilde{\kappa}$ is also a constant. This one-class SVM thus corresponds to the MEB problem (1), in which φ is replaced by the nonlinear map $\tilde{\varphi}$ satisfying $\tilde{\varphi}(\mathbf{z}_i)'\tilde{\varphi}(\mathbf{z}_j) = \tilde{k}(\mathbf{z}_i, \mathbf{z}_j)$. From the Karush-Kuhn-Tucker (KKT) conditions, we can recover $\mathbf{w} = \sum_{i=1}^m \alpha_i\varphi(\mathbf{x}_i)$ and $\xi_i = \frac{\alpha_i}{C}$, and $\rho = \mathbf{w}'\varphi(\mathbf{x}_i) + \frac{\alpha_i}{C}$ from any support vector $\mathbf{x}_i$.

3.2.2 Two-Class L2-SVM

Given a training set $\{\mathbf{z}_i = (\mathbf{x}_i, y_i)\}_{i=1}^m$ with $y_i \in \{-1, 1\}$, the primal of the two-class L2-SVM is

$$\begin{aligned} \min_{\mathbf{w},b,\rho,\xi_i} \quad & \|\mathbf{w}\|^2 + b^2 - 2\rho + C\sum_{i=1}^{m} \xi_i^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}'\varphi(\mathbf{x}_i) + b) \ge \rho - \xi_i. \end{aligned} \quad (8)$$

The corresponding dual is

$$\begin{aligned} & \max_{\mathbf{0}\le\boldsymbol{\alpha}} -\boldsymbol{\alpha}'\left(\mathbf{K}\odot\mathbf{y}\mathbf{y}' + \mathbf{y}\mathbf{y}' + \frac{1}{C}\mathbf{I}\right)\boldsymbol{\alpha} \;:\; \boldsymbol{\alpha}'\mathbf{1} = 1 \\ = \;& \max -\boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha} \;:\; \mathbf{0} \le \boldsymbol{\alpha}, \;\; \boldsymbol{\alpha}'\mathbf{1} = 1, \end{aligned} \quad (9)$$

where $\odot$ denotes the Hadamard product, $\mathbf{y} = [y_1, \ldots, y_m]'$ and $\tilde{\mathbf{K}} = [\tilde{k}(\mathbf{z}_i, \mathbf{z}_j)]$ with

$$\tilde{k}(\mathbf{z}_i, \mathbf{z}_j) = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + y_i y_j + \frac{\delta_{ij}}{C}, \quad (10)$$

involving both input and label information. Again, this is of the form in (5), with $\tilde{k}(\mathbf{z}, \mathbf{z}) = \kappa + 1 + \frac{1}{C} \equiv \tilde{\kappa}$, a constant. Again, we can recover

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \varphi(\mathbf{x}_i), \quad b = \sum_{i=1}^{m} \alpha_i y_i, \quad \xi_i = \frac{\alpha_i}{C}, \quad (11)$$

from the optimal $\boldsymbol{\alpha}$ and $\rho = y_i(\mathbf{w}'\varphi(\mathbf{x}_i) + b) + \frac{\alpha_i}{C}$ from any support vector $\mathbf{z}_i$. Note that all the support vectors of this L2-SVM, including those defining the margin and those that are misclassified, now reside on the surface of the ball in the feature space induced by $\tilde{k}$. A similar relationship connecting one-class classification and binary classification is also described in (Schölkopf et al., 2001).

[2] In this case, it can be shown that the hard (soft) margin SVDD yields identical solution as the hard (soft) margin one-class SVM (Schölkopf et al., 2001). Moreover, the weight $\mathbf{w}$ in the one-class SVM solution is equal to the center $\mathbf{c}$ in the SVDD solution.

4 Core Vector Machine (CVM)

After formulating the kernel method as a MEB problem, we obtain a transformed kernel $\tilde{k}$, together with the associated feature space $\tilde{\mathcal{F}}$, mapping $\tilde{\varphi}$ and constant $\tilde{\kappa} = \tilde{k}(\mathbf{z}, \mathbf{z})$. To solve this kernel-induced MEB problem, we adopt the approximation algorithm[3] described in the proof of Theorem 2.2 in (Bădoiu & Clarkson, 2002). As mentioned in Section 2, the idea is to incrementally expand the ball by including the point furthest away from the current center. In the following, we denote the core-set, the ball's center and radius at the tth iteration by $\mathcal{S}_t$, $\mathbf{c}_t$ and R_t respectively. Also, the center and radius of a ball B are denoted by $\mathbf{c}_B$ and r_B. Given an $\epsilon > 0$, the CVM then works as follows:

1. Initialize $\mathcal{S}_0$, $\mathbf{c}_0$ and R_0.
2. Terminate if there is no $\tilde{\varphi}(\mathbf{z})$ (where $\mathbf{z}$ is a training point) falling outside the $(1+\epsilon)$-ball $B(\mathbf{c}_t, (1+\epsilon)R_t)$.
3. Find $\mathbf{z}$ such that $\tilde{\varphi}(\mathbf{z})$ is furthest away from $\mathbf{c}_t$. Set $S_{t+1} = \mathcal{S}_t \cup \{\mathbf{z}\}$.
4. Find the new MEB$(\mathcal{S}_{t+1})$ from (5) and set $\mathbf{c}_{t+1} = \mathbf{c}_{\text{MEB}(\mathcal{S}_{t+1})}$ and $R_{t+1} = r_{\text{MEB}(\mathcal{S}_{t+1})}$ using (3).
5. Increment t by 1 and go back to step 2.

In the sequel, points that are added to the core-set will be called *core vectors*. Details of each of the above steps will be described in Section 4.1. Despite its simplicity, CVM has an approximation guarantee (Section 4.2) and also provably small time and space complexities (Section 4.3).

4.1 Detailed Procedure

4.1.1 Initialization

Bădoiu and Clarkson (2002) simply used an arbitrary point $\mathbf{z} \in \mathcal{S}$ to initialize $S_0 = \{\mathbf{z}\}$. However, a good initialization may lead to fewer updates and so we follow the scheme in (Kumar et al., 2003). We start with an arbitrary point $\mathbf{z} \in \mathcal{S}$ and find $\mathbf{z}_a \in \mathcal{S}$ that is furthest away from $\mathbf{z}$ in the feature space $\tilde{\mathcal{F}}$. Then, we find another point $\mathbf{z}_b \in \mathcal{S}$ that is furthest away from $\mathbf{z}_a$ in $\tilde{\mathcal{F}}$. The initial core-set is then set to be $\mathcal{S}_0 = \{\mathbf{z}_a, \mathbf{z}_b\}$. Obviously, MEB$(\mathcal{S}_0)$ (in $\tilde{\mathcal{F}}$) has center $\mathbf{c}_0 = \frac{1}{2}(\tilde{\varphi}(\mathbf{z}_a) + \tilde{\varphi}(\mathbf{z}_b))$ On using (3), we thus have $\alpha_a = \alpha_b = \frac{1}{2}$ and all the other α_i's are zero. The initial radius is $R_0 = \frac{1}{2}\|\tilde{\varphi}(\mathbf{z}_a) - \tilde{\varphi}(\mathbf{z}_b)\| = \frac{1}{2}\sqrt{\|\tilde{\varphi}(\mathbf{z}_a)\|^2 + \|\tilde{\varphi}(\mathbf{z}_b)\|^2 - 2\tilde{\varphi}(\mathbf{z}_a)'\tilde{\varphi}(\mathbf{z}_b)} = \frac{1}{2}\sqrt{2\tilde{\kappa} - 2\tilde{k}(\mathbf{z}_a, \mathbf{z}_b)}$.

In a classification problem, one may further require $\mathbf{z}_a$ and $\mathbf{z}_b$ to come from different classes. On using (10), R_0 then becomes $\frac{1}{2}\sqrt{2\left(\kappa + 2 + \frac{1}{C}\right) + 2k(\mathbf{x}_a, \mathbf{x}_b)}$. As κ and C are constants, choosing the pair $(\mathbf{x}_a, \mathbf{x}_b)$ that maximizes R_0 is then equivalent to choosing the closest pair belonging to opposing classes, which is also the heuristic used in initializing the SimpleSVM (Vishwanathan et al., 2003).

[3] A similar algorithm is also described in (Kumar et al., 2003).

4.1.2 Distance Computations

Steps 2 and 3 involve computing $\|\mathbf{c}_t - \tilde{\varphi}(\mathbf{z}_\ell)\|$ for $\mathbf{z}_\ell \in \mathcal{S}$. Now,

$$\begin{aligned}&\|\mathbf{c}_t - \tilde{\varphi}(\mathbf{z}_\ell)\|^2 \qquad (12)\\ &= \sum_{\mathbf{z}_i, \mathbf{z}_j \in \mathcal{S}_t} \alpha_i \alpha_j \tilde{k}(\mathbf{z}_i, \mathbf{z}_j) - 2\sum_{\mathbf{z}_i \in \mathcal{S}_t} \alpha_i \tilde{k}(\mathbf{z}_i, \mathbf{z}_\ell) + \tilde{k}(\mathbf{z}_\ell, \mathbf{z}_\ell),\end{aligned}$$

on using (3). Hence, computations are based on kernel evaluations instead of the explicit $\tilde{\varphi}(\mathbf{z}_i)$'s, which may be infinite-dimensional. Note that, in contrast, existing MEB algorithms only consider finite-dimensional spaces.

However, in the feature space, $\mathbf{c}_t$ cannot be obtained as an explicit point but rather as a convex combination of (at most) $|\mathcal{S}_t|$ $\tilde{\varphi}(\mathbf{z}_i)$'s. Computing (12) for all m training points takes $O(|\mathcal{S}_t|^2 + m|\mathcal{S}_t|) = O(m|\mathcal{S}_t|)$ time at the tth iteration. This becomes very expensive when m is large. Here, we use the probabilistic speedup method in (Smola & Schölkopf, 2000). The idea is to randomly sample a sufficiently large subset $\mathcal{S}'$ from $\mathcal{S}$, and then take the point in $\mathcal{S}'$ that is furthest away from $\mathbf{c}_t$ as the approximate furthest point over $\mathcal{S}$. As shown in (Smola & Schölkopf, 2000), by using a small random sample of, say, size 59, the furthest point obtained from $\mathcal{S}'$ is with probability 0.95 among the furthest 5% of points from the whole $\mathcal{S}$. Instead of taking $O(m|\mathcal{S}_t|)$ time, this randomized method only takes $O(|\mathcal{S}_t|^2 + |\mathcal{S}_t|) = O(|\mathcal{S}_t|^2)$ time, which is much faster as $|\mathcal{S}_t| \ll m$. This trick can also be used in initialization.

4.1.3 Adding the Furthest Point

Points outside MEB$(\mathcal{S}_t)$ have zero α_i's (Section 4.1.1) and so violate the KKT conditions of the dual problem. As in (Osuna et al., 1997), one can simply add any such violating point to $\mathcal{S}_t$. Our step 3, however, takes a greedy approach by including the point furthest away from the current center. In the classification case[4] (Section 3.2.2), we have

$$\begin{aligned}&\arg \max_{\mathbf{z}_\ell \notin B(\mathbf{c}_t, (1+\epsilon)R_t)} \|\mathbf{c}_t - \tilde{\varphi}(\mathbf{z}_\ell)\|^2\\ &= \arg \min_{\mathbf{z}_\ell \notin B(\mathbf{c}_t, (1+\epsilon)R_t)} \sum_{\mathbf{z}_i \in \mathcal{S}_t} \alpha_i y_i y_\ell (k(\mathbf{x}_i, \mathbf{x}_\ell) + 1)\\ &= \arg \min_{\mathbf{z}_\ell \notin B(\mathbf{c}_t, (1+\epsilon)R_t)} y_\ell(\mathbf{w}'\varphi(\mathbf{x}_\ell) + b), \qquad (13)\end{aligned}$$

on using (10), (11) and (12). Hence, (13) chooses the *worst* violating pattern corresponding to the constraint (8). Also, as the dual objective in (9) has gradient $-2\tilde{\mathbf{K}}\boldsymbol{\alpha}$, so for a pattern ℓ currently outside the ball

$$\begin{aligned}(\tilde{\mathbf{K}}\boldsymbol{\alpha})_\ell &= \sum_{i=1}^{m} \alpha_i \left(y_i y_\ell k(\mathbf{x}_i, \mathbf{x}_\ell) + y_i y_\ell + \frac{\delta_{i\ell}}{C}\right)\\ &= y_\ell(\mathbf{w}'\varphi(\mathbf{x}_\ell) + b),\end{aligned}$$

[4] The case for one-class classification (Section 3.2.1) is similar.

on using (10), (11) and $\alpha_\ell = 0$. Thus, the pattern chosen in (13) also makes the most progress towards maximizing the dual objective. This subset selection heuristic has been commonly used by various decomposition algorithms (e.g., (Chang & Lin, 2004; Joachims, 1999; Platt, 1999)).

4.1.4 Finding the MEB

At each iteration of step 4, we find the MEB by using the QP formulation in Section 3.2. As the size $|\mathcal{S}_t|$ of the core-set is much smaller than m in practice (Section 5), the computational complexity of each QP sub-problem is much lower than solving the whole QP. Besides, as only one core vector is added at each iteration, efficient rank-one update procedures (Cauwenberghs & Poggio, 2001; Vishwanathan et al., 2003) can also be used. The cost then becomes quadratic rather than cubic. In the current implementation (Section 5), we use SMO. As only one point is added each time, the new QP is just a slight perturbation of the original. Hence, by using the MEB solution obtained from the previous iteration as starting point (*warm start*), SMO can often converge in a small number of iterations.

4.2 Convergence to (Approximate) Optimality

First, consider $\epsilon = 0$. The proof in (Bădoiu & Clarkson, 2002) does not apply as it requires $\epsilon > 0$. Nevertheless, as the number of core vectors increases by one at each iteration and the training set size is finite, so CVM must terminate in a finite number (say, τ) of iterations, With $\epsilon = 0$, MEB($\mathcal{S}_\tau$) is an enclosing ball for all the points on termination. Because $\mathcal{S}_\tau$ is a subset of the whole training set and the MEB of a subset cannot be larger than the MEB of the whole set. Hence, MEB($\mathcal{S}_\tau$) must also be the exact MEB of the whole ($\tilde{\varphi}$-transformed) training set. In other words, when $\epsilon = 0$, CVM outputs the *exact* solution of the kernel problem.

Now, consider $\epsilon > 0$. Assume that the algorithm terminates at the τth iteration, then

$$R_\tau \leq r_{\text{MEB}(\mathcal{S})} \leq (1+\epsilon) R_\tau \tag{14}$$

by definition. Recall that the optimal primal objective p^* of the kernel problem in Section 3.2.1 (or 3.2.2) is equal to the optimal dual objective d_2^* in (7) (or (9)), which in turn is related to the optimal dual objective $d_1^* = r^2_{\text{MEB}(\mathcal{S})}$ in (2) by (6). Together with (14), we can then bound p^* as

$$R_\tau^2 \leq p^* + \tilde{\kappa} \leq (1+\epsilon)^2 R_\tau^2. \tag{15}$$

Hence, $\max\left(\frac{R_\tau^2}{p^*+\tilde{\kappa}}, \frac{p^*+\tilde{\kappa}}{R_\tau^2}\right) \leq (1+\epsilon)^2$ and thus CVM is an $(1+\epsilon)^2$-approximation algorithm. This also holds with high probability when probabilistic speedup is used.

As mentioned in Section 1, practical SVM implementations also output approximated solutions only. Typically, a parameter similar to our ϵ is required at termination. For example, in SMO and SVM^{light} (Joachims, 1999), training stops when the KKT conditions are fulfilled within ϵ. Experience with these softwares indicate that near-optimal solutions are often good enough in practical applications. Moreover, it can also be shown that when the CVM terminates, all the points satisfy loose KKT conditions as in SMO and SVM^{light}.

4.3 Time and Space Complexities

Existing decomposition algorithms cannot guarantee the number of iterations and consequently the overall time complexity (Chang & Lin, 2004). In this Section, we show how this can be obtained for CVM. In the following, we assume that a plain QP implementation, which takes $O(m^3)$ time and $O(m^2)$ space for m patterns, is used for the MEB sub-problem in Section 4.1.4. Moreover, we assume that each kernel evaluation takes constant time.

As proved in (Bădoiu & Clarkson, 2002), CVM converges in at most $2/\epsilon$ iterations. In other words, the total number of iterations, and consequently the size of the final core-set, are of $\tau = O(1/\epsilon)$. In practice, it has often been observed that the size of the core-set is much smaller than this worst-case theoretical upper bound (Kumar et al., 2003). This will also be corroborated by our experiments in Section 5.

Consider first the case where probabilistic speedup is not used in Section 4.1.2. As only one core vector is added at each iteration, $|S_t| = t + 2$. Initialization takes $O(m)$ time while distance computations in steps 2 and 3 take $O((t+2)^2 + tm) = O(t^2 + tm)$ time. Finding the MEB in step 4 takes $O((t+2)^3) = O(t^3)$ time, and the other operations take constant time. Hence, the tth iteration takes $O(tm + t^3)$ time, and the overall time for $\tau = O(1/\epsilon)$ iterations is

$$\sum_{t=1}^{\tau} O(tm + t^3) = O(\tau^2 m + \tau^4) = O\left(\frac{m}{\epsilon^2} + \frac{1}{\epsilon^4}\right),$$

which is *linear* in m for a fixed ϵ.

As for space[5], since only the core vectors are involved in the QP, the space complexity for the tth iteration is $O(|\mathcal{S}_t|^2)$. As $\tau = O(1/\epsilon)$, the space complexity for the whole procedure is $O(1/\epsilon^2)$, which is *independent* of m for a fixed ϵ.

On the other hand, when probabilistic speedup is used, initialization only takes $O(1)$ time while distance computations in steps 2 and 3 take $O((t+2)^2) = O(t^2)$ time. Time for the other operations remains the same. Hence, tth iteration takes $O(t^3)$ time and the whole procedure takes

$$\sum_{t=1}^{\tau} O(t^3) = O(\tau^4) = O\left(\frac{1}{\epsilon^4}\right).$$

[5] As the patterns may be stored out of core, we ignore the $O(m)$ space required for storing the m patterns.

For a fixed ϵ, it is thus *constant*, independent of m. The space complexity, which depends only on the number of iterations τ, is still $O(1/\epsilon^2)$.

If more efficient QP solvers were used in the MEB subproblem of Section 4.1.4, both the time and space complexities can be further improved. For example, with SMO, the space complexity for the tth iteration is reduced to $O(|\mathcal{S}_t|)$ and that for the whole procedure driven down to $O(1/\epsilon)$.

Note that when ϵ decreases, the CVM solution becomes closer to the exact optimal solution, but at the expense of higher time and space complexities. Such a tradeoff between efficiency and approximation quality is typical of all approximation schemes. Morever, be cautioned that the O-notation is used for studying the asymptotic efficiency of algorithms. As we are interested on handling very large data sets, an algorithm that is asymptotically more efficient (in time and space) will be the best choice. However, on smaller problems, this may be outperformed by algorithms that are not as efficient asymptotically. These will be demonstrated experimentally in Section 5.

5 Experiments

In this Section, we implement the two-class L2-SVM in Section 3.2.2 and illustrate the scaling behavior of CVM (in C++) on both toy and real-world data sets. For comparison, we also run the following SVM implementations[6]:

1. L2-SVM: LIBSVM implementation (in C++);
2. L2-SVM: LSVM implementation (in MATLAB), with low-rank approximation (Fine & Scheinberg, 2001) of the kernel matrix added;
3. L2-SVM: RSVM (Lee & Mangasarian, 2001) implementation (in MATLAB). The RSVM addresses the scale-up issue by solving a smaller optimization problem that involves a random $\bar{m} \times m$ rectangular subset of the kernel matrix. Here, $\bar{m}$ is set to 10% of m;
4. L1-SVM: LIBSVM implementation (in C++);
5. L1-SVM: SimpleSVM (Vishwanathan et al., 2003) implementation (in MATLAB).

Parameters are used in their default settings unless otherwise specified. All experiments are performed on a 3.2GHz Pentium–4 machine with 512M RAM, running Windows XP. Since our focus is on nonlinear kernels, we use the Gaussian kernel $k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2/\beta)$, with $\beta = \frac{1}{m^2}\sum_{i,j=1}^{m} \|\mathbf{x}_i - \mathbf{x}_j\|^2$.

Our CVM implementation is adapted from LIBSVM, and uses SMO for each QP sub-problem in Section 4.1.4. As in LIBSVM, our CVM also uses caching (with the same cache size as in the other LIBSVM implementations above) and stores all training patterns in main memory. For simplicity, shrinking is not used in our current CVM implementation. Moreover, we employ probabilistic speedup (Section 4.1.2) and set $\epsilon = 10^{-6}$ in all the experiments. As in other decomposition methods, the use of a very stringent stopping criterion is not necessary in practice. Preliminary studies show that $\epsilon = 10^{-6}$ is acceptable for most tasks. Using an even smaller ϵ does not show improved generalization performance, but may increase the training time unnecessarily.

5.1 Checkerboard Data

We first experiment on the 4×4 checkerboard data used by Lee and Mangasarian (2001) for evaluating large-scale SVM implementations. We use training sets with a maximum of 1 million points and 2000 independent points for testing. Of course, this problem does not need so many points for training, but it is convenient for illustrating the scaling properties. Experimentally, L2-SVM with low rank approximation does not yield satisfactory performance on this data set, and so its result is not reported here. RSVM, on the other hand, has to keep a rectangular kernel matrix of size $\bar{m} \times m$ and cannot be run on our machine when m exceeds 10K. Similarly, the SimpleSVM has to store the kernel matrix of the active set, and runs into storage problem when m exceeds 30K.

As can be seen from Figure 2, CVM is as accurate as the others. Besides, it is much faster[7] and produces far fewer support vectors (which implies faster testing) on large data sets. In particular, one million patterns can be processed in under 13s. On the other hand, for relatively small training sets, with less than 10K patterns, LIBSVM is faster. This, however, is to be expected as LIBSVM uses more sophisticated heuristics and so will be more efficient on small-to-medium sized data sets. Figure 2(b) also shows the core-set size, which can be seen to be small and its curve basically overlaps with that of the CVM. Thus, almost all the core vectors are useful support vectors. Moreover, it also confirms our theoretical findings that both time and space are constant w.r.t. the training set size, when it is large enough.

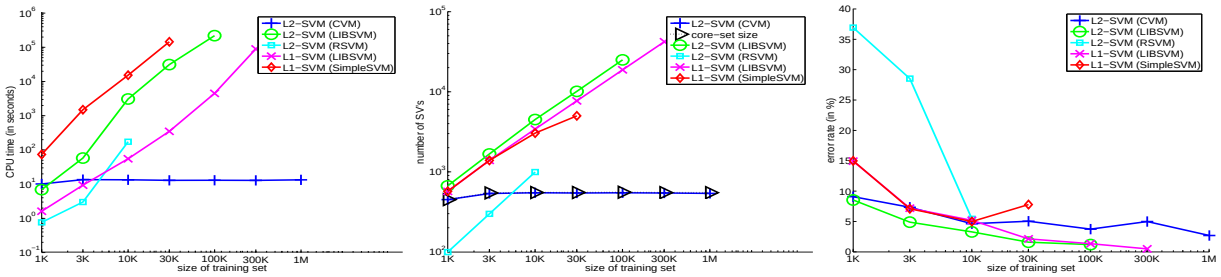
5.2 Forest Cover Type Data[8]

This data set has been used for large scale SVM training by Collobert et al. (2002). Following (Collobert et al.,

[6] Our CVM implementation can be downloaded from http://www.cs.ust.hk/∼jamesk/cvm.zip. LIBSVM can be downloaded from http://www.csie.ntu.edu.tw/∼cjlin/libsvm/; LSVM from http://www.cs.wisc.edu/dmi/lsvm; and SimpleSVM from http://asi.insa-rouen.fr/∼gloosli/. Moreover, we followed http://www.csie.ntu.edu.tw/∼cjlin/libsvm/faq.html in adapting the LIBSVM package for L2-SVM.

[7] As some implementations are in MATLAB, so not all the CPU time measurements can be directly compared. However, it is still useful to note the constant scaling exhibited by the CVM and its speed advantage over other C++ implementations, when the data set is large.

[8] http://kdd.ics.uci.edu/databases/covertype/covertype.html



(a) CPU time. (b) number of SV's. (c) testing error.

Figure 2: Results on the checkerboard data set (Except for the CVM, all the other implementations have to terminate early because of not enough memory and / or the training time is too long). Note that the CPU time, number of support vectors, and size of the training set are in log scale.

2002), we aim at separating class 2 from the other classes. $1\% - 90\%$ of the whole data set (with a maximum of 522,911 patterns) are used for training while the remaining are used for testing. We set $\beta = 10000$ for the Gaussian kernel. Preliminary studies show that the number of support vectors is over ten thousands. Consequently, RSVM and SimpleSVM cannot be run on our machine. Similarly, for low rank approximation, preliminary studies show that over thousands of basis vectors are required for a good approximation. Therefore, only the two LIBSVM implementations will be compared with the CVM here.

Figure 3 shows that CVM is, again, as accurate as the others. Note that when the training set is small, more training patterns bring in additional information useful for classification and so the number of core vectors increases with training set size. However, after processing around 100K patterns, both the time and space requirements of CVM begin to exhibit a constant scaling with the training set size. With hindsight, one might simply sample 100K training patterns and hope to obtain comparable results[9]. However, for satisfactory classification performance, different problems require samples of different sizes and CVM has the important advantage that the required sample size does not have to be pre-specified. Without such prior knowledge, random sampling gives poor testing results, as has been demonstrated in (Lee & Mangasarian, 2001).

5.3 Relatively Small Data Sets: UCI Adult Data[10]

Following (Platt, 1999), we use training sets with up to 32,562 patterns. As can be seen in Figure 4, CVM is still among the most accurate methods. However, as this data set is relatively small, more training patterns do carry more classification information. Hence, as discussed in Section 5.2, the number of iterations, the core set size and consequently the CPU time all increase with the number of training patterns. From another perspective, recall that the worst case core-set size is $2/\epsilon$, independent of m (Section 4.3). For the value of $\epsilon = 10^{-6}$ used here, $2/\epsilon = 2 \times 10^6$. Although we have seen that the actual size of the core-set is often much smaller than this worst case value, however, when $m \ll 2/\epsilon$, the number of core vectors can still be dependent on m. Moreover, as has been observed in Section 5.1, the CVM is slower than the more sophisticated LIBSVM on processing these smaller data sets.

[9] In fact, we tried both LIBSVM implementations on a random sample of 100K training patterns, but their testing accuracies are inferior to that of CVM.

[10] http://research.microsoft.com/users/jplatt/smo.html

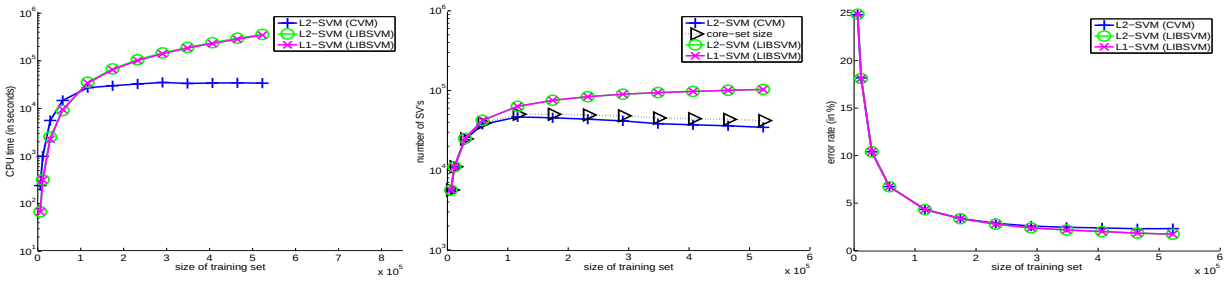
6 Conclusion

In this paper, we exploit the "approximateness" in SVM implementations. We formulate kernel methods as equivalent MEB problems, and then obtain provably approximately optimal solutions efficiently with the use of core-sets. The proposed CVM procedure is simple, and does not require sophisticated heuristics as in other decomposition methods. Moreover, despite its simplicity, CVM has small asymptotic time and space complexities. In particular, for a fixed ϵ, its asymptotic time complexity is *linear* in the training set size m while its space complexity is *independent* of m. When probabilistic speedup is used, it even has *constant* asymptotic time and space complexities for a fixed ϵ, independent of the training set size m. Experimentally, on large data sets, it is much faster and produce far fewer support vectors (and thus faster testing) than existing methods. On the other hand, on relatively small data sets where $m \ll 2/\epsilon$, SMO can be faster. CVM can also be used for other kernel methods such as support vector regression, and details will be reported elsewhere.

References

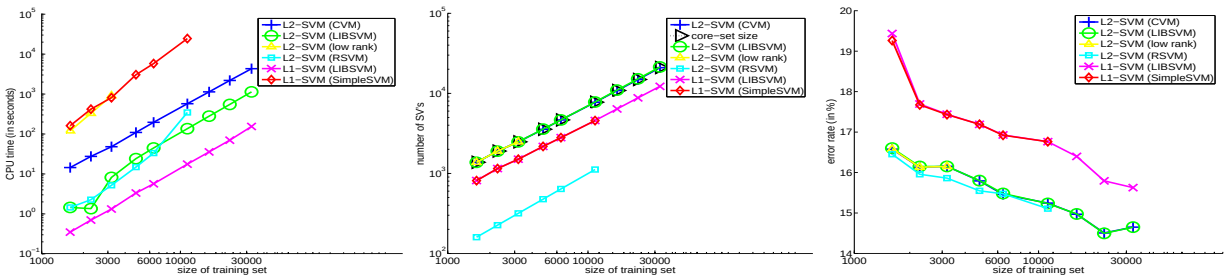
Bădoiu, M., & Clarkson, K. (2002). Optimal core-sets for balls. *DIMACS Workshop on Computational Geometry*.

Cauwenberghs, G., & Poggio, T. (2001). Incremental and decremental support vector machine learning. *Advances in Neural Information Processing Systems 13*. Cambridge, MA: MIT Press.



(a) CPU time. (b) number of SV's. (c) testing error.

Figure 3: Results on the forest cover type data set. Note that the y-axes in Figures 3(a) and 3(b) are in log scale.



(a) CPU time. (b) number of SV's. (c) testing error.

Figure 4: Results on the UCI adult data set (The other implementations have to terminate early because of not enough memory and/or training time is too long). Note that the CPU time, number of SV's and size of training set are in log scale.

Chang, C.-C., & Lin, C.-J. (2004). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chapelle, O., Vapnik, V., Bousquet, O., & Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, *46*, 131–159.

Collobert, R., Bengio, S., & Bengio, Y. (2002). A parallel mixture of SVMs for very large scale problems. *Neural Computation*, *14*, 1105–1114.

Fine, S., & Scheinberg, K. (2001). Efficient SVM training using low-rank kernel representation. *Journal of Machine Learning Research*, *2*, 243–264.

Garey, M., & Johnson, D. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman.

Joachims, T. (1999). Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges and A. Smola (Eds.), *Advances in kernel methods – Support vector learning*, 169–184. Cambridge, MA: MIT Press.

Kumar, P., Mitchell, J., & Yildirim, A. (2003). Approximate minimum enclosing balls in high dimensions using core-sets. *ACM Journal of Experimental Algorithmics*, *8*.

Lee, Y.-J., & Mangasarian, O. (2001). RSVM: Reduced support vector machines. *Proceeding of the First SIAM International Conference on Data Mining*.

Lin, K.-M., & Lin, C.-J. (2003). A study on reduced support vector machines. *IEEE Transactions on Neural Networks*, *14*, 1449–1459.

Mangasarian, O., & Musicant, D. (2001). Lagrangian support vector machines. *Journal of Machine Learning Research*, *1*, 161–177.

Osuna, E., Freund, R., & Girosi, F. (1997). Training support vector machines: an application to face detection. *Proceedings of Computer Vision and Pattern Recognition* (pp. 130–136). San Juan, Puerto Rico.

Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges and A. Smola (Eds.), *Advances in kernel methods – support vector learning*, 185–208. Cambridge, MA: MIT Press.

Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A., & Williamson, R. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, *13*, 1443–1471.

Smola, A., & Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 911–918). Standord, CA, USA.

Smola, A., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*, 199–222.

Tax, D., & Duin, R. (1999). Support vector domain description. *Pattern Recognition Letters*, *20*, 1191–1199.

Vishwanathan, S., Smola, A., & Murty, M. (2003). SimpleSVM. *Proceedings of the Twentieth International Conference on Machine Learning* (pp. 760–767). Washington, D.C., USA.

Williams, C., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems 13*. Cambridge, MA: MIT Press.

Streaming Feature Selection using IIC

Lyle H. Ungar and Jing Zhou
Computer and Information Science
University of Pennsylvania, Philadelphia, PA 19104
ungar, jingzhou@seas.upenn.edu

Dean P. Foster and Bob A. Stine
Statistics Department
University of Pennsylvania, Philadelphia, PA 19104
foster, stine@wharton.upenn.edu

Abstract

In Streaming Feature Selection (SFS), new features are sequentially considered for addition to a predictive model. When the space of potential features is large, SFS offers many advantages over methods in which all features are assumed to be known in advance. Features can be generated dynamically, focusing the search for new features on promising subspaces, and overfitting can be controlled by dynamically adjusting the threshold for adding features to the model. We present a new, adaptive complexity penalty, the Information Investing Criterion (IIC), which uses an efficient coding of features added, and not added, to the model to dynamically adjust the threshold on the entropy reduction required for adding a new feature. Streaming Feature Selection with IIC gives strong guarantees against overfitting. In contrast, standard penalty methods such as BIC or RIC always drastically over- or under-fit in the limit of infinite numbers of non-predictive features. Empirical results show that SFS is competitive with much more compute-intensive feature selection methods.

1 Introduction

In many problems, one has a fixed set of observations from which a vast, or even infinite stream of features can be generated to build predictive models. The large number of potentially predictive features may come from transformations of, and interactions between, a smaller initial set of features. For example, most commercial statistical software offers the ability to do stepwise regression using all feature interactions (e.g., products of pairs of features, or all products containing three variables). Pairwise interactions are important and, along with data transformations, can rapidly create large data sets. For example in a bankruptcy prediction problem described below, considering interactions between the 365 original features led to a set of over 67,000 potential features, of which about 40 proved significant.

The features may also come from more complex feature generation algorithms. For example, Statistical Relational Learning (SRL) methods often generate tens or hundreds of thousands of potentially predictive features. SRL and related methods "crawl" through a database or other relational structure and generate features by building increasingly complex compound relations [1] . For example, when building a model to predict the journal in which an article will be published, potentially predictive features include the words in the target article itself, the words in the articles cited by the target article, the words in articles that cite articles written by the authors of the target article, and so forth. Traversing such relational structures can easily generates millions of features, since there are many words, authors, and journals. Current modeling techniques, however, are ill equipped to deal with problems of learning from, say, a million potential features for each of a hundred thousand observations. A hundred billion numbers do not fit easily into memory on most contemporary computers. More importantly, CPU is fast relative to memory, and being more so.

When building models from potentially enormous sets of features, it is desirable to interleave the process of feature generation with that of feature testing in order to avoid even generating features which are less likely to be useful. One may want to only consider interaction terms in a regression if at least one of the component terms has proven predictive. One may want to only search farther in those branches of a refinement graph in inductive logic programming (ILP) which contain terms that have proven predictive – as is, indeed, done in ILP. Building predictive models from such large, complex data sets requires careful control to avoid over-fitting, particularly when there are many more features than observations. Standard statistical and machine learning methods such as SVMs, maximum entropy methods and neural networks generally assume that all features ("predictors") are known in advance. They then use regu-

larization or features selection to avoid overfitting.

This paper focuses on penalty-based feature selection methods for problems in which a small number of predictive features are to be selected from a large set of potential features. We will compare, in the context of streaming feature selection, the widely used BIC penalty method with RIC, a more recent penalty method, and with the new Information Investing Criterion (IIC), which this paper introduces.

BIC can be understood in an information theoretic sense as consisting of a code (specifying the parameters in the model) and the compressed data (describing the errors in the predictions made by the model). Each zero parameter (feature not included in the model) is coded with one bit, and each non-zero parameter is coded with $1 + \frac{1}{2}\log(n)$ bits, where n is the number of observations used. (All logs are base 2.) Recalling that the log likelihood of the data given a model gives the number of bits to code the model error, leads to the BIC criterion for feature selection: accept a new feature x_i only if the change in log likelihood from adding the feature is greater than $\frac{1}{2}\log(n)$, i.e. if $\log(P(Y|\hat{Y}_i)) - \log(P(Y|\hat{Y}_{-i})) > \frac{1}{2}\log(n)$. BIC is equivalent to a Minimum Description Length (MDL)[2] criterion if the number of features considered, p is much less than the number of observations, n. Howver, BIC is not a valid code for $p \gg n$.

The Risk Inflation Criterion (RIC) [3, 4] gives another, much more stringent criterion for feature selection, which controls the minimax risk. RIC chooses a set of features from the potential feature pool so that the loss of the resulting model is within a factor of $\log(p)$ of the loss of the best such model. In essence, RIC behaves like a Bonferroni rule, in which a threshold for feature inclusion is selected so that *the set of all features* will only have a small chance of containing a "false" feature. This is highly conservative, and does often not produce optimal out of sample prediction accuracies.

The Information Investing Criterion (IIC) introduced in this paper is an alternative MDL-style coding which, unlike BIC and RIC, is adaptive. Information investing does not require knowing the number of potential predictors in advance, yet still has provable bounds on overfitting. IIC's performance is never much worse than BIC or RIC, and for the types of problems we are interested in, where there are far more potential features than observations, it often gives vastly superior performance.

The assumptions behind penalty methods such as BIC and RIC are not met when a fixed number of features are to be selected from an arbitrarily large set of potentially predictive features. Inclusion rules such as AIC and BIC, which are not a function of p, the number of possible features to be considered for inclusion in the model, inevitably overfit as p becomes large. When presented with a continuous sequence of features that are random noise, any selection procedure that generates false positives at a fixed rate, such as AIC or BIC, will select infinitely many of these random features as predictors. Inclusion rules such as RIC (Bonferroni) which *are* a function of p under-fit as p becomes large. Any such method that reduces the chance of including each feature based on the total number of features, p, to be considered will end up not adding any features in the limit as $p \to \infty$.

The solution to this dilemma is to sequentially consider a $stream$ of features for model inclusion and use a method which incrementally adjusts the criterion for including new features in the model depending on the history of addition (or non-addition) of features seen so far. We argue for Streaming Feature Selection (SFS), where as each additional feature is observed, it is tested for inclusion in the model and then either included or discarded. Streaming feature selection offers many advantages over the traditional approach of stepwise selection from a fixed set of features. In stepwise regression, all features are considered for addition to the model, the best one is selected, and then all remaining features considered, etc. At every iteration, almost all features are tested for addition to the model. This requires having a finite set of features specified in advance, and requires looking at each feature many times. Stepwise feature selection is widely used with penalty methods such as AIC and BIC, but we will show below that streaming feature selection often gives competitive performance, while allowing much greater flexibility in dynamically controlling feature generation. Using streams of features has other benefits. Since most features will not be included in the models, they can be discarded soon after generation, thus reducing data storage requirement and allowing the solution of larger problems than can be tackled using standard machine learning algorithms such as support vector machines (SVMs) or neural networks which assume that all potentially predictive features are known *a priori*.

2 Streaming feature selection

The goal of streaming feature selection is to pick useful predictors from an offered sequence of features. For a fixed set of observations, new features (predictors) are considered sequentially, and the decision to include or discard each feature is made at the time it is provided. SFS can be used with a variety of different machine learning methods; all it requires from the machine learner is that it take features sequentially and produce an estimate of the change in entropy (log-likelihood) in the model. A wide range of classical statistical methods can be used off-the-shelf, such as linear or logistic regression, or extensions such as generalized linear methods and estimating equations. SFS works particularly well with modeling methods than can efficiently add additional features and with adaptive penalty methods such as IIC.

```
Initialize
    i = 1, wealth = w_0 bits, model = {}
Do forever
    x ← get_new_feature()
    ϵ ← wealth/2i
    bits_saved ← entropy_reduction(x, ϵ, model)
    if(bits_saved > w_Δ)
        wealth ← wealth + w_Δ
        add_feature(x, model) // add x to the model
    else
        wealth ← wealth − ϵ
    i ← i + 1
```

Figure 1: Information-investing algorithm

SFS dynamically adjusts the threshold, w_Δ, on the entropy reduction needed for a new variable to enter the model.[1] The threshold, w_Δ, is adjusted using the wealth, w_i, which represents the number of bits currently available for overfitting. Wealth starts at an initial value w_0 specifying the number of bits by which one is willing to risk increasing the description length. It is increased by w_Δ each time a variable (feature) is added to the model, since the variable is guaranteed to save at least w_Δ bits, and decreased by ϵ each time a variable is not added to the model,reflecting (as described below) the cost of coding the fact that the feature was not added.

The algorithm is given in Figure 1. ϵ specifies how many bits are available to code a variable. The $bits\_saved$ by adding a feature to the model is the net entropy reduction from adding x to the model: the reduction in the model error minus the cost of coding the coefficient, β, associated with x and the cost of indicating that the variable is to be added to the model. Different codings can be used for the coefficients, for example $\frac{1}{2}log(n)$ bits (or, for a very approximate coding 3 bits) to code each nonzero coefficient, and e.g. -$\log(\epsilon)$ bits to code that x is to be added to the model. (Since ϵ is the number of bits available to code a spurious feature, the probability of the next feature being "useful." is $1 - e^{-\epsilon} = 1 - (1 - \epsilon + O(\epsilon^2)) \approx \epsilon$, and the cost in bits of coding that that the feature is useful is roughly -$\log(\epsilon)$ bits.) If β, the coefficient of x, has an associated t-statistic, then adding x to the model reduces the entropy of the model by $\frac{1}{2}t^2log(e)$. (The log(e) converts the t^2 to bits.)

If the feature x reduces entropy sufficiently to be worth adding to the model, then the wealth is incremented by a fixed amount, w_Δ. If the feature x is not added to the model, the wealth is decreased by the cost of coding the variable's absence, which by an argument similar for that used above is $-\log(1-\epsilon)$, which, for small ϵ, is approximately ϵ.

[1] A very similar SFS algorithm, which we call α-$investing$, can be written that dynamically adjusts the criterion for adding a new feature to a model based on the p-value of the feature under consideration.

2.1 Guarantees against overfitting

One sense in which SFS is guaranteed not to over-fit, is that on average, the sum of the total description length plus the wealth will never increase. Since the wealth is strictly positive, this guarantees that the total description length can never increase by more than the current wealth. Since when a feature is added to the model we increase the wealth less than the description length decreases, the description length plus wealth tends to decrease, providing on average better models.

SFS also provides another, much more subtle, guarantee against overfitting. For the case of "hard" problems, where the coefficients to be estimated are just barely distinguishable above the noise, the cost of adding a "false" feature is comparable to the benefit of adding a true features. This is a property of using a so-called *testimator*. A testimator tests for significance and then estimates by the usual estimator if it is significant, and estimates by zero otherwise. If a variable has a true coefficient of zero, then when it is falsely included, it will be biased by about $t_\alpha SE$, where t_α is the critical value used for testing significance, and SE is the standard error of the coefficient. On the other hand, the hardest to detect coefficients will have a coefficient of about $t_\alpha SE$. Hence leaving them out will bias their estimated value by about the same amount, namely $t_\alpha SE$. We can thus get optimal test error by adding as many features as possible while not exceeding a specified ratio of false to true features added to the model.[2]

SFS using the IIC coding (described below) allows us, for any valid coding, to bound in expectation the ratio of incorrect features added to correct features added, and thus to minimize the expected test error by adding as many features as possible subject to controlling that ratio.

Theorem
Let M_i be the number of correct variables included in the model, let N_i be the number of spurious variables (those with true coefficient zero) included and w_i be the wealth, all at iteration i, and let $w_\Delta < 1/4$ be a user selected value. Then if the algorithm in Figure 1 is modified so that it never bids more than $1/2$ it will have the property that:

$$E(N_i) < 4w_\Delta E(M_i) + 4w_0.$$

[2] This is very similar to controlling the False Discovery Rate (FDR) [5], the number of features incorrectly included in the model divided by the total number of features included in the model, which has become popular in recent years. In the regime that we are working, correctly adding a feature always reduces both the FDR and the out-of-sample error, and incorrectly adding a feature always increases both FDR and error.

Proof Sketch
The proof relies on the fact that $S_i \equiv N_i - 4w_\Delta M_i + 4w_i$ is a super-martingale, namely at each time period the conditional expectation of $S_i - S_{i-1}$ is negative. We will show that S_i is a super-martingale by considering the cases when the variable is or is not in the true model and is or is not added to the estimated model.

	$\beta_i = 0$	$\beta_i \neq 0$
use zero	$\Delta M_i = 0, \Delta N_i = 0$	$\Delta M_i = 0, \Delta N_i = 0$
add variable	$\Delta M_i = 0, \Delta N_i = 1$	$\Delta M_i = 1, \Delta N_i = 0$

We can write the change in S_i as:

$$\begin{aligned} \Delta S_i &\equiv S_i - S_{i-1} \\ &= \Delta N_i - 4w_\Delta \Delta M_i + 4\Delta w_i \end{aligned}$$

If $\beta_i \neq 0$, then $\Delta N_i = 0$. Thus,

$$\Delta S_i = -\epsilon_i(1 - \Delta M_i) \leq 0,$$

where ϵ_i is the amount bid at time i. On the other hand, if $\beta_i = 0$, then $\Delta M_i = 0$. So,

$$\begin{aligned} \Delta S_i &= \Delta N_i + 4\Delta w_i \\ &= \Delta N_i + 4w_\Delta \Delta N_i - 4\epsilon_i(1 - \Delta N_i) \\ &= \Delta N_i(1 + 4w_\Delta + \epsilon_i) - 4\epsilon_i. \end{aligned}$$

By bounds from information theory, we see that $E(\Delta N_i) \leq \epsilon_i$. Also by assumption, $4w_\Delta \leq 1$ and $\epsilon_i \leq 1/2$. Hence $E(\Delta S_i) \leq 4\epsilon_i - 4\epsilon_i = 0$. Thus, S_i is a super-martingale.

Using the weaker fact that for super-martingales: $E(S_i) \leq E(S_{i-1})$, we see that $E(S_i) \leq S_0$. But since we start out with $N_i = 0$, and $M_i = 0$, $S_0 = 4w_0$. Since $w_i > 0$ by construction, we see that $E(N_i - 4w_\Delta M_i) < 4w_0$. □

When $w_\Delta = \frac{1}{4}$, this reduces to $E(N_i) < E(M_i) + 4w_0$. The expected number of spurious variables added is thus no more than $4w_0$ greater than the expected number of correct variables added.

As described above, if we add as many features as possible subject to meeting such a constraint on spurious to true features added, we will minimize the expected test error. The selection of ϵ_i as $w_i/2i$ gives the slowest possible decrease in wealth such that all wealth is used; i.e., so that as many features as possible are included in the model without systematically over-fitting. More formally:

Theorem
Computing ϵ_i as $w_i/2i$ gives the slowest possible decrease in wealth such that $\lim_{i\to\infty} w_i = 0$.

Proof Sketch
Define $\delta_i = \epsilon_i/w_i$ to be the fraction of wealth invested at time i. If no features are added to the model, wealth at time i is $w_i = \Pi_i(1 - \delta_i)$. If we pass to the limit to generate w_∞, we have $w_\infty = \Pi_i(1 - \delta_i) = e^{\sum \log(1-\delta_i)} = e^{-\sum \delta_i + O(\delta_i^2)}$. Thus, $w_\infty = 0$ iff $\sum \delta_i$ is infinite.

Thus if we let δ_i go to zero faster than $1/i$, say $i^{-1-\gamma}$ where $\gamma > 0$ then $w_\infty > 0$ and we have wealth that we never use.

2.2 IIC and its coding scheme

A key question is what coding scheme to use in determining the entropy reduction. We describe here an "optimal" coding scheme which leads to the information investing algorithm described in Figure 1. Our goal is to find a (legitimate) coding scheme which, given a "bid," ϵ, specifying how many bits are available to code a variable, will guarantee the highest probability of adding the variable to the model. The key idea is that log(probability) and bits are equivalent. This equivalence allows us to think in terms of distributions and thus to compute codes which handle fractions of a bit. We show in this section that given any actual distribution $\tilde{f}_\beta$ of the coefficients, we can produce a coding corresponding to a modified distribution f_β which uniformly dominates the coding implied by $\tilde{f}_\beta$.

Assume, for simplicity, that we increase the wealth by one bit when a variable x_i with coefficient β_i is added. Thus, when x_i is added $\log(p(x_i \text{ is a "true" variable}) / p(x_i \text{ is a "false" variable})) > 1$ bit; i.e. the log-likelihood decreases by more than one bit. Let f_{β_i} be the distribution implied by the coding scheme for t_{β_i} if we add x_i and $f_0(t_{\beta_i})$ be the normal distribution (the null model in which x_i should not be added). The coding saves enough bits to justify adding a variable whenever $f_{\beta_i}(t_{\beta_i}) \geq 2 * f_0(t_{\beta_i})$. This happens with probability $\alpha_i \equiv p_0(\{t_{\beta_i} : f_{\beta_i}(t_{\beta_i}) \geq 2 * f_0(t_{\beta_i})\})$ under the null (α_i is thus the area under the tails of the null distribution.)

There is no reason to have $f_{\beta_i}(t_{\beta_i}) \gg 2 * f_0(t_{\beta_i})$ in the tails, since this would "waste" probability or bits. Hence the optimal coding corresponds to $f_\beta(t_{\beta_i}) = 2 * f_0(t_{\beta_i})$ for all the variables that are likely to be added. Using all of the remaining probability mass (or equivalently, making the coding "Kraft tight") dictates the coding for the case when the variable is not likely to be added. The most efficient coding to use is thus:

$$\begin{cases} f_\beta(t_{\beta_i}) = 2f_0(t_{\beta_i}) & \text{if } |t_{\beta_i}| > t_{\alpha_i} \\ f_\beta(t_{\beta_i}) = \frac{1-2*\alpha_i}{1-\alpha_i} f_0(t_{\beta_i}) & \text{otherwise} \end{cases}$$

and the corresponding cost in bits is:

$$\begin{cases} \log(f_\beta(t_{\beta_i})/f_0(t_{\beta_i})) = \log(2) = 1 \text{ bit} & \text{if } |t_{\beta_i}| > t_{\alpha_i} \\ \log(f_\beta(t_{\beta_i})/f_0(t_{\beta_i}) = & \\ \qquad \log(\frac{1-2\alpha_i}{1-\alpha_i}) \approx -\alpha_i \text{ bits} & \text{otherwise} \end{cases}$$

Figure 2 shows the distribution $f_\beta(t_(\beta_i))$, with the probability mass transfered away from the center, where features are not added, out to the tails, where features are added.

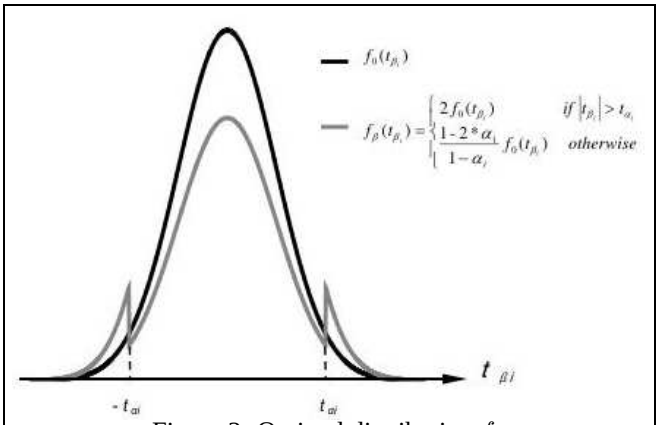


Figure 2: Optimal distribution f_β

		BIC	RIC	SFS
streaming	**features**	39.3	7.1	5.4
	error	3.21	2.88	3.16
stepwise	**features**	199	11.1	–
	error	3.89	2.40	–

Table 1. BIC overfits for $p \gg n$. Average number of features selected and out-of-sample error. n = 200 observations, p = 1,000 features, q = 10 true features in model Synthetic data: $x \sim N(0,1)$ y: linear in x with noise $\sigma^2 = 5$. A perfect model would give test error of 2.236, the error of the null model is 3.873. The results are an average over 20 runs, and reported errors have an uncertainty of around 0.02.

The above equations been derived assuming that 1 bit is added to the wealth. It can be generalized to add w_Δ bits to the wealth each time a variable is added to the model. Then, when a variable is added to the model the probability of it being "true" should be 2^{w_Δ} times that of it being "false", and all of the 2's in the above equations are replaced with 2^{w_Δ}.

3 Experimental Results

To further illustrate the method, we evaluate SFS on a synthetic data set for which the correct answers are known and on a larger, real data set of bankruptcy prediction. The base synthetic data set contains 200 observations each of 1,000 features, of which 10 are predictive. We generate the features independently from a normal distribution, $N(0,1)$, with the true model being the sum of the ten predictors plus noise, $N(0,5)$. The artificially simple structure of the data allows us to easily see which feature selection methods are adding spurious variables or failing to find variables that should be in the model.

The results are presented in Table 1. As expected, BIC overfits, although less badly when streaming is used rather than the stepwise selection procedure. RIC gives performance superior to SFS in this particular case ($q = 10$) but it fails badly when its assumptions (q small) are violated, as shown in Table 2. Stepwise regression using RIC does better here than the streaming version. However, using standard code from R, the stepwise regression was *much* slower than the streaming regression, to the point where running stepwise regression on data sets with tens of thousands of features was not possible.

	BIC	RIC	SFS
features	89.3	25.5	61.5
error	6.24	9.57	7.60

Table 2. RIC underfits for $q \gg 1$. Same parameters as Table 1 (streaming) except n = 1,000, q = 100 features in data and $\sigma^2 = 15$.

One might hope that adding more spurious features to the end of a feature stream would not severely harm an algorithm's performance. However, BIC, since its penalty is not a function of p, will add even more spurious variables (if BIC haven't already added a feature for every observation!). RIC (or Bonferroni) puts a harsher penalty as p gets large, adding fewer and fewer features. As Table 3 shows, SFS is clearly the superior method when the true features occur early in the feature stream. SFS continues to occasionally add features to the model, which would be good if there were predictive features later in the stream, but does not lead to much overfitting when there are no such features.

It is often the case that large numbers of features are generated, with the best ones tending to be earlier in the sequence. Such feature streams are generated when one searches over interactions and transformations, as in the bankruptcy example presented below. Similar feature streams arise when one computes features at many length scales, as for face or object recognition in machine vision. Another example is Structural Relational Learning (SRL), where potential features are generated by searching the space of logic queries or relational database queries.

We have used the CiteSeer data set, which contains about 500,000 papers, 100,000 "constants" (words, authors, and journals), and around ten different relations (including author, venue, cites, has-word, institution, download) to predict which journal a given paper will be published in, or which papers it will cite. Predictions using CiteSeer benefit from the generation of rich sets of features, and depending on the exact task, SFS gives out-of-sample errors comparable to, or several percentage points below those from non-adaptive techniques [6]. Learning in SRL methods such as Structural Generalized Linear Regression (SGLR) [6] benefit from efficient integration of feature generation and selection; as each feature is tested for possible inclusion in the model, the results are fed back to the feature genera-

	p	1,000	10,000	100k	1M
BIC	**features**	39.3	199	199	199
	false pos.	29.5	189	189	189
	error	3.21	4.45	4.45	4.45
RIC	**features**	7.1	3.8	1.2	0.9
	false pos.	0.1	0.5	0.1	0.2
	error	2.88	3.49	3.77	3.91
SFS	**features**	5.4	5.4	5.7	5.7
	false pos.	0.3	0.5	0.8	0.8
	error	3.16	3.30	3.29	3.29

Table 3. Effect of adding spurious features Same parameters as Table 1 except that additional spurious features have been added after the first 1,000 features. "false pos." indicates the average number of features incorrectly added. (average over 10 runs)

tor, which can then use this information to determine which further features to generate. Since generating the features from these databases takes CPU days, avoiding generating features is important both for computational as well as for statistical efficiency methods.

We also tested a slight modification of SFS on a problem of predicting personal bankruptcies[7]. The data set is highly un-balanced, containing 2,244 bankruptcy events and hundreds of thousands of non-bankruptcy observations. The real world loss function for predicting bankruptcy is quite asymmetric: the cost of predicting a bankruptcy when none occurs is much higher than the cost of failing to predict a bankruptcy when one does occur. We call the ratio of these two costs ρ.

We compared Streaming Feature Selection against boosted C4.5, doing 5-fold cross-validation, where each pass of the cross-validation uses 100,000 non-bankruptcies and about one fifth of the bankruptcies. SFS was run once, and then the out-of-sample costs were estimated for each cost ratio, ρ using the predicted probability of bankruptcy. C4.5 was run separately for each value of ρ.

ρ	199	99	19	6	4	1
C.45 cost	132	76	18.6	7.2	5.09	1.45
SFS cost	61	41	15.3	6.9	5.02	1.54

Table 4. Loss as a function of the loss ratio, ρ, for boosted C4.5 and for SFS

Table 4 shows that for low cost ratios, the two methods give very similar results, but at higher cost ratios, SFS gives around half the loss of C4.5. Using AIC, one would expect over 1,000 variables to be falsely included in the model, based on the fact that an f-statistic-based penalty of 2 corresponds to a t-statistic of $\sqrt{2}$ which is a wildly generous threshold when considering 67,000 features. BIC also massively overfits, although less severely.

4 Alternate feature selection methods

Recent developments in statistical variable selection take into account the size of the feature space, but only allow for finite, fixed feature spaces, and do not support sequential (or streaming) feature selection. The risk inflation criterion (RIC) produces a model that possesses a type of competitive predictive optimality [4, 3]. RIC chooses a set of features from the potential feature pool so that the loss of the resulting model is within a factor of $\log(p)$ of the loss of the best such model. In essence, RIC behaves like a Bonferroni rule [3]. Each time a predictor is considered, there is a chance that it will enter the model even if it is merely noise. In other words, the tested null hypothesis is that the proposed feature does not improve the prediction of the model. Doing a formal test generates a p-value for this null hypothesis. Suppose we only add this predictor if its p-value is less than α_j when we consider the jth predictor. Then the Bonferroni rule keeps the chance of adding even one extraneous predictor to less than, say, 0.05 by constraining $\sum \alpha_j \leq 0.05$.

Bonferroni methods like RIC are conservative, limiting the ability of a model to add factors that improve its predictive accuracy. The connection of RIC to α-spending rules leads to a more powerful alternative. An α-spending rule is a multiple comparison procedure that bounds its cumulative type 1 error rate at a small level, say 5%. For example, suppose one has to test the p hypotheses $H_1, H_2, \ldots, H_p$. If we test the first using level α_Δ, the second using level α_2 and so forth with $\sum_j \alpha_j = 0.05$, then we have only a 5% chance of falsely rejecting one of the p hypotheses. If we associate each hypothesis with the claim that a predictor adds to value to a regression, then we are back in the situation of a Bonferroni rule for variable selection. Bonferroni methods and RIC simply fix $\alpha_j = \alpha/p$ for each test.

Alternative multiple comparison procedures control a different property. Rather than control the cumulative α (also known as the family wide error rate), these control the so-called false discovery rate [5]. Control of the false discovery rate at 5% implies that at most 5% of the rejected hypotheses are false positives. In variable selection, this implies that of the included predictors, at most 5% degrade the accuracy of the model. The Benjamini-Hochberg method for controlling the false discovery rate suggests the α-spending method for keeping the false discovery rate below α: Order the p-values of the independents tests of $H_1, H_2, \ldots, H_p$ so that $p_1 \leq p_2 \leq \cdots p_p$. Now find the largest p-value for which $p_k \leq \alpha/(p-k)$ and reject all H_i for $i \leq k$. Thus, if the smallest p-value $p_1 \leq \alpha/p$, it is rejected. Rather than compare the second largest p-value to the RIC/Bonferroni threshold α/p, reject H_2 if $p_2 \leq 2\alpha/p$.

There have been many papers that looked at procedures of this sort for use in variable selection from an FDR perspective [8], an empirical Bayesian perspective [9, 10], an information theoretical perspective [11] or simply a data mining perspective [7]. But all of these require knowing the entire list of possible variables ahead of time. Further, most of them assume that the variables are orthogonal and hence tacitly assume that $p < n$.

We are currently exploring a way of doing SFS that uses what we call α-investing instead of IIC. In SFS using IIC, we keep track of the number of bits saved and use these bits to invest in future variables. Whereas in α-investing the medium of exchange is the accumulation of α that has yet to be spent. When a significant variable is found, the α-spending account goes up, but when a variable is found to be insignificant, the account decreases. Though the α-investing rule sounds like it might be close to Benjamini-Hochberg's FDR procedure described above, it turns out to be fairly different. In particular, the Benjamini-Hochberg method fails as p gets large; it is a batch-oriented procedure. But the α-investing shares with IIC the property of not needing to know p ahead of time and hence being able to handle a potentially infinite stream of predictors.

5 Summary

A variety of machine learning algorithms have been developed for online learning where *observations* are sequentially added. Algorithms such as SFS which are online in the *features* being used are much less common. For some problems, all predictors are known in advance, and a large fraction of them are predictive. In such cases, regularization or smoothing methods work well and streaming feature selection does not make sense. For other problems, selecting a small number of features gives a much stronger model than trying to smooth across all potential features. (See [12, 13] for a range of feature selection problems and approaches.) For example, in predicting what journal an article will be published in, we find that roughly 10-20 of the 80,000 features we examine are selected [14]. For the problems in citation prediction and bankruptcy prediction that we have looked at, generating potential features (e.g. by querying a database or by computing transformations or combinations of the raw features) takes orders of magnitude more time than the machine learning done by streaming feature selection. Thus, the flexibility that SFS provides to dynamically decide which features to generate and add to the feature stream provides potentially large savings in computation.

Streaming feature selection can be done using any penalty method such as AIC, BIC or RIC, but is functions best when using a method such as IIC which adapts the penalty as a function of what features have been seen and added at each point. The widely used BIC criterion is only valid in the limit as the number of observations n goes to infinity while the number of features p remains small. The more modern RIC assumes that n and p are large but that the number of true variables in the model is close to one. Unlike BIC and RIC, IIC works for all values of p and n, and for any $q \ll p$. The results presented in this paper are for "hard" problems, in which the coefficients are close to the limit of being detectable above the noise. For easy problems, where the signal to noise ratio is high, all methods tend to work reasonably well. For problems which have a mix of easy and hard coefficients, the SFS algorithm can be modified to make multiple passes, first "investing" a relatively small number of bits to find the easy features, and then using the algorithm as described above to find the hard features.

Key to the guarantee that IIC works for widely varying values of n, p and q is the use of an adaptive penalty to control the ratio of correct ("true") to incorrect ("false") features by using an information theoretic coding to adjust the threshold on the entropy reduction necessary for adding a variable to the model. Streaming Feature Selection with IIC is extremely easily to implement on top of any algorithm which incrementally considers features for addition and calculates their entropy reduction or p-value. For linear and logistic regression, we have found that SFS can easily handle millions of features.

References

[1] S. Dzeroski and N. Lavrac. *Relational Data Mining*. Springer-Verlag, 2001.

[2] Jorma Rissanen. Hypothesis selection and testing by the mdl principle. *The Computer Journal*, 42:260–269, 1999.

[3] D. P. Foster and E. I. George. The risk inflation criterion for multiple regression. *Annals of Statistics*, 22:1947–1975, 1994.

[4] D. L. Donoho and I. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81:425–455, 1994.

[5] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society*, Series B(57):289–300, 1995.

[6] A. Popescul and L. H. Ungar. Cluster-based concept invention for statistical relational learning. In *Proc. Conference Knowledge Discovery and Data Mining (KDD-2004)*, 2004.

[7] D. P. Foster and R. A. Stine. Variable selection in data mining: Building a predictive model for bankruptcy. *Journal of the American Statistical Association (JASA)*, 2004. 303-313.

[8] Felix Abramovich, Y. Benjamini, D. Donoho, and Ian Johnstone. Adapting to unknown sparsity by controlling the false discovery rate. Technical Report 2000–19, Dept. of Statistics, Stanford University, Stanford, CA, 2000.

[9] E. I. George and D. P. Foster. Calibration and empirical bayes variable selection. *Biometrika*, 87:731–747, 2000.

[10] I. M. Johnstone and B. W. Silverman. Needles and straw in haystacks: Empirical bayes estimates of possibly sparse sequences. *Annals of Statistics*, 32:1594–1649, 2004.

[11] D. P. Foster and R. A. Stine. Adaptive variable selection competes with Bayes experts. Submitted for publication, 2004.

[12] In *JMLR Special Issue on Variable Selection*. Journal of Machine Learning Research (JMLR), 2003.

[13] In *NIPS 2003 workshop on feature extraction*, 2003.

[14] A. Popescul and L. H. Ungar. Structural logistic regression for link analysis. In *KDD Workshop on Multi-Relational Data Mining*, 2003.

Defensive Forecasting

Vladimir Vovk[*]
vovk@cs.rhul.ac.uk
http://vovk.net

Akimichi Takemura[†]
takemura@stat.t.u-tokyo.ac.jp
http://www.e.u-tokyo.ac.jp/~takemura

Glenn Shafer[‡*]
gshafer@andromeda.rutgers.edu
http://glennshafer.com

Abstract

We consider how to make probability forecasts of binary labels. Our main mathematical result is that for any continuous gambling strategy used for detecting disagreement between the forecasts and the actual labels, there exists a forecasting strategy whose forecasts are ideal as far as this gambling strategy is concerned. A forecasting strategy obtained in this way from a gambling strategy demonstrating a strong law of large numbers is simplified and studied empirically.

1 INTRODUCTION

Probability forecasting can be thought of as a game between two players, Forecaster and Reality:

FOR $n = 1, 2, \dots$:
Reality announces $x_n \in \mathbf{X}$.
Forecaster announces $p_n \in [0, 1]$.
Reality announces $y_n \in \{0, 1\}$.

On each round, Forecaster predicts Reality's move y_n chosen from the *label space*, always taken to be $\{0, 1\}$ in this paper. His move, the *probability forecast* p_n, can be interpreted as the probability he attaches to the event $y_n = 1$. To help Forecaster, Reality presents him with an *object* x_n at the beginning of the round; x_n are chosen from an *object space* $\mathbf{X}$.

Forecaster's goal is to produce p_n that agree with the observed y_n. Various results of probability theory, in particular limit theorems (such as the weak and strong laws of large numbers, the law of the iterated logarithm, and the central limit theorem) and large-deviation inequalities (such as Hoeffding's inequality), describe different aspects of agreement between p_n and y_n. For example, according to the strong law of large numbers, we expect that

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} (y_i - p_i) = 0. \tag{1}$$

Such results will be called *laws of probability* and the existing body of laws of probability will be called *classical probability theory*.

In §2, following [12], we formalize Forecaster's goal by adding a third player, Skeptic, who is allowed to gamble at the odds given by Forecaster's probabilities. We state a result from [14] and [12] suggesting that Skeptic's gambling strategies can be used as tests of agreement between p_n and y_n and that all tests of agreement between p_n and y_n can be expressed as Skeptic's gambling strategies. Therefore, the forecasting protocol with Skeptic provides an alternative way of stating laws of probability.

As demonstrated in [12], many standard proof techniques developed in classical probability theory can be translated into continuous strategies for Skeptic. In §3 we show that for any continuous strategy $\mathcal{S}$ for Skeptic there exists a strategy $\mathcal{F}$ for Forecaster such that $\mathcal{S}$ does not detect any disagreement between the y_n and the p_n produced by $\mathcal{F}$. This result is a "meta-theorem" that allows one to move from laws of probability to forecasting algorithms: as soon as a law of probability is expressed as a continuous strategy for Skeptic, we have a forecasting algorithm that guarantees that this law will hold; there are no assumptions about Reality, who may play adversarially.

Our meta-theorem is of any interest only if one can find sufficiently interesting laws of probability (expressed as gambling strategies) that can serve as its input. In §4

[*]Computer Learning Research Centre, Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, England.

[†]Department of Mathematical Informatics, Graduate School of Information Science and Technology, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan.

[‡]Rutgers Business School—Newark and New Brunswick, 180 University Avenue, Newark, NJ 07102, USA.

we apply it to the important properties of unbiasedness in the large and small of the forecasts p_n ((1) is an asymptotic version of the former). The resulting forecasting strategy is automatically unbiased, no matter what data $x_1, y_1, x_2, y_2, \ldots$ is observed.

In §5 we simplify the algorithm obtained in §4 and demonstrate its performance on some artificially generated data sets.

2 THE GAMBLING FRAMEWORK FOR TESTING PROBABILITY FORECASTS

Skeptic is allowed to bet at the odds defined by Forecaster's probabilities, and he refutes the probabilities if he multiplies his capital manyfold. This is formalized as a perfect-information game in which Skeptic plays against a team composed of Forecaster and Reality:

BINARY FORECASTING GAME I

Players: Reality, Forecaster, Skeptic

Protocol:

$\mathcal{K}_0 := 1.$
FOR $n = 1, 2, \ldots:$
  Reality announces $x_n \in \mathbf{X}$.
  Forecaster announces $p_n \in [0, 1]$.
  Skeptic announces $s_n \in \mathbb{R}$.
  Reality announces $y_n \in \{0, 1\}$.
  $\mathcal{K}_n := \mathcal{K}_{n-1} + s_n(y_n - p_n)$.

Restriction on Skeptic: Skeptic must choose the s_n so that his capital is always nonnegative ($\mathcal{K}_n \geq 0$ for all n) no matter how the other players move.

This is a perfect-information protocol; the players move in the order indicated, and each player sees the other player's moves as they are made. It specifies both an initial value for Skeptic's capital ($\mathcal{K}_0 = 1$) and a lower bound on its subsequent values ($\mathcal{K}_n \geq 0$).

Our interpretation, which will be called the *testing interpretation*, of Binary Forecasting Game I is that $\mathcal{K}_n$ measures the degree to which Skeptic has shown Forecaster to do a bad job of predicting y_i, $i = 1, \ldots, n$.

2.1 VALIDITY AND UNIVERSALITY OF THE TESTING INTERPRETATION

As explained in [12], the testing interpretation is valid and universal in an important sense. Let us assume, for simplicity, that objects are absent (formally, that $|\mathbf{X}| = 1$). In the case where Forecaster starts from a probability measure P on $\{0, 1\}^\infty$ and obtains his forecasts $p_n \in [0, 1]$ as conditional probabilities under P that $y_n = 1$ given $y_1, \ldots, y_{n-1}$, we have a standard way of testing P and, therefore, p_n: choose an event $A \subseteq \{0, 1\}^\infty$ (the *critical region*) with a small $P(A)$ and reject P if A happens. The testing interpretation satisfies the following two properties:

Validity Suppose Skeptic's strategy is measurable and p_n are obtained from P; $\mathcal{K}_n$ then form a nonnegative martingale w.r. to P. According to Doob's inequality [14, 3], for any positive constant C, $\sup_n \mathcal{K}_n \geq C$ with P-probability at most $1/C$. (If Forecaster is doing a bad job according to the testing interpretation, he is also doing a bad job from the standard point of view.)

Universality According to Ville's theorem ([12], §8.5), for any positive constant ϵ and any event $A \subseteq \{0, 1\}^\infty$ such that $P(A) < \epsilon$, Skeptic has a measurable strategy that ensures $\liminf_{n\to\infty} \mathcal{K}_n > 1/\epsilon$ whenever A happens, provided p_n are computed from P. (If Forecaster is doing a bad job according to the standard point of view, he is also doing a bad job according to the testing interpretation.) In the case $P(A) = 0$, Skeptic actually has a measurable strategy that ensures $\lim_{n\to\infty} \mathcal{K}_n = \infty$ on A.

The universality of the gambling scenario of Binary Forecasting Game I is its most important advantage over von Mises's gambling scenario based on subsequence selection; it was discovered by Ville [14].

2.2 CONTINUITY OF GAMBLING STRATEGIES

In [12] we constructed Skeptic's strategies that made him rich when the statement of any of several key laws of probability theory was violated. The constructions were explicit and lead to continuous gambling strategies. We conjecture that every natural result of classical probability theory leads to a continuous strategy for Skeptic.

3 DEFEATING SKEPTIC

In this section we prove the main (albeit very simple) mathematical result of this paper: for any continuous strategy for Skeptic there exists a strategy for Forecaster that does not allow Skeptic's capital to grow, regardless of what Reality is doing. Actually, our result will be even stronger: we will have Skeptic announce his strategy for each round before Forecaster's move on that round rather than making him announce his full strategy at the beginning of the game, and we will drop the restriction on Skeptic. Therefore, we con-

sider the following perfect-information game that pits Forecaster against the two other players:

BINARY FORECASTING GAME II

Players: Reality, Forecaster, Skeptic

Protocol:

$\mathcal{K}_0 := 1$.
FOR $n = 1, 2, \ldots$:
  Reality announces $x_n \in \mathbf{X}$.
  Skeptic announces continuous $S_n : [0,1] \to \mathbb{R}$.
  Forecaster announces $p_n \in [0,1]$.
  Reality announces $y_n \in \{0,1\}$.
  $\mathcal{K}_n := \mathcal{K}_{n-1} + S_n(p_n)(y_n - p_n)$.

Theorem 1 *Forecaster has a strategy in Binary Forecasting Game II that ensures $\mathcal{K}_0 \geq \mathcal{K}_1 \geq \mathcal{K}_2 \geq \cdots$.*

Proof Forecaster can use the following strategy to ensure $\mathcal{K}_0 \geq \mathcal{K}_1 \geq \cdots$:

- if the function $S_n(p)$ takes the value 0, choose p_n so that $S_n(p_n) = 0$;
- if S_n is always positive, take $p_n := 1$;
- if S_n is always negative, take $p_n := 0$. ∎

4 EXAMPLES OF GAMBLING STRATEGIES

In this section we discuss strategies for Forecaster obtained by Theorem 1 from different strategies for Skeptic; the former will be called *defensive forecasting strategies*. There are many results of classical probability theory that we could use, but we will concentrate on the simple strategy described in [12], p. 69, for proving the strong law of large numbers.

If $S_n(p) = S_n$ does not depend on p, the strategy from the proof of Theorem 1 makes Forecaster choose

$$p_n := \begin{cases} 0 & \text{if } S_n < 0 \\ 1 & \text{if } S_n > 0 \\ 0 \text{ or } 1 & \text{if } S_n = 0. \end{cases}$$

The basic procedure described in [12] (p. 69) is as follows. Let $\epsilon \in (0, 0.5]$ be a small number (expressing our tolerance to violations of the strong law of large numbers). In Binary Forecasting Game I, Skeptic can ensure that

$$\sup_n \mathcal{K}_n < \infty \implies \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^n (y_i - p_i) \leq \epsilon \quad (2)$$

using the strategy $s_n = s_n^\epsilon := \epsilon \mathcal{K}_{n-1}$. Indeed, since

$$\mathcal{K}_n = \prod_{i=1}^n (1 + \epsilon(y_i - p_i)),$$

on the paths where $\mathcal{K}_n$ is bounded we have

$$\prod_{i=1}^n (1 + \epsilon(y_i - p_i)) \leq C,$$

$$\sum_{i=1}^n \ln(1 + \epsilon(y_i - p_i)) \leq \ln C,$$

$$\epsilon \sum_{i=1}^n (y_i - p_i) - \epsilon^2 \sum_{i=1}^n (y_i - p_i)^2 \leq \ln C,$$

$$\epsilon \sum_{i=1}^n (y_i - p_i) \leq \ln C + \epsilon^2 n,$$

$$\frac{1}{n} \sum_{i=1}^n (y_i - p_i) \leq \frac{\ln C}{\epsilon n} + \epsilon$$

(we have used the fact that $\ln(1+t) \geq t - t^2$ when $|t| \leq 0.5$). If Skeptic wants to ensure

$$\sup_n \mathcal{K}_n < \infty \implies -\epsilon \leq \liminf_{n\to\infty} \frac{1}{n} \sum_{i=1}^n (y_i - p_i) \leq \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^n (y_i - p_i) \leq \epsilon,$$

he can use the strategy $s_n := (s_n^\epsilon + s_n^{-\epsilon})/2$, and if he wants to ensure

$$\sup_n \mathcal{K}_n < \infty \implies \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^n (y_i - p_i) = 0, \quad (3)$$

he can use a convex mixture of $(s_n^\epsilon + s_n^{-\epsilon})/2$ over a sequence of ϵ converging to zero. There are also standard ways of strengthening (3) to

$$\liminf_{n\to\infty} \mathcal{K}_n < \infty \implies \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^n (y_i - p_i) = 0;$$

for details, see [12].

In the rest of this section we will draw on the excellent survey [2]. We will see how Forecaster defeats increasingly sophisticated strategies for Skeptic.

4.1 UNBIASEDNESS IN THE LARGE

Following Murphy and Epstein [7], we say that Forecaster is *unbiased in the large* if (1) holds. Let us first consider the one-sided relaxed version of this property

$$\limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^n (y_i - p_i) \leq \epsilon. \quad (4)$$

The strategy for Skeptic described above, $S_n(p) := \epsilon\mathcal{K}_n$, leads to Forecaster always choosing $p_n := 1$; (4) is then satisfied in a trivial way.

Forecaster's strategy corresponding to the two-sided version

$$-\epsilon \le \liminf_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n}(y_i - p_i) \le \limsup_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n}(y_i - p_i) \le \epsilon \quad (5)$$

is not much more reasonable. Indeed, it can be represented as follows. The initial capital 1 is split evenly between two accounts, and Skeptic gambles with the two accounts separately. If at the outset of round n the capital on the first account is $\mathcal{K}^1_{n-1}$ and the capital on the second account is $\mathcal{K}^2_{n-1}$, Skeptic plays $s^1_n := \epsilon\mathcal{K}^1_{n-1}$ with the first account and $s^2_n := -\epsilon\mathcal{K}^2_{n-1}$ with the second account; his total move is

$$S_n(p) := \epsilon\mathcal{K}^1_{n-1} - \epsilon\mathcal{K}^2_{n-1} = \epsilon\left(\prod_{i=1}^{n-1}(1+\epsilon(y_i-p_i)) - \prod_{i=1}^{n-1}(1+\epsilon(p_i-y_i))\right).$$

Therefore, Forecaster's move is $p_n := 1$ if

$$\sum_{i=1}^{n-1}\ln(1+\epsilon(y_i-p_i)) > \sum_{i=1}^{n-1}\ln(1+\epsilon(p_i-y_i)),$$

$p_n := 0$ if

$$\sum_{i=1}^{n-1}\ln(1+\epsilon(y_i-p_i)) < \sum_{i=1}^{n-1}\ln(1+\epsilon(p_i-y_i)),$$

and p_n can be chosen arbitrarily in the case of equality. The limiting form of this strategy as $\epsilon \to 0$ is: Forecaster's move is $p_n := 1$ if

$$\sum_{i=1}^{n-1}(y_i - p_i) > 0,$$

$p_n := 0$ if

$$\sum_{i=1}^{n-1}(y_i - p_i) < 0,$$

and p_n can be chosen arbitrarily in the case of equality.

We can see that unbiasedness in the large does not lead to interesting forecasts: Forecaster fulfils his task too well. In the one-sided case (4), he always chooses $p_n := 1$ making

$$\sum_{i=1}^{n}(y_i - p_i)$$

as small as possible. In the two-sided case (5) with $\epsilon \to 0$, he manages to guarantee that

$$\left|\sum_{i=1}^{n}(y_i - p_i)\right| \le 1. \quad (6)$$

His goals are achieved with categorical forecasts, $p_n \in \{0,1\}$.

In the rest of this section we consider the more interesting case where $S_n(p)$ depends on p.

4.2 UNBIASEDNESS IN THE SMALL

We now consider a subtler requirement that forecasts should satisfy, which we introduce informally. We say that the forecasts p_n are *unbiased in the small* (or reliable, or valid, or well calibrated) if, for any $p^* \in [0,1]$,

$$\frac{\sum_{i=1,\dots,n:p_i\approx p^*} y_i}{\sum_{i=1,\dots,n:p_i\approx p^*} 1} \approx p^* \quad (7)$$

provided $\sum_{i=1,\dots,n:p_i\approx p^*} 1$ is not too small.

Let us first consider just one value for p^*. Instead of the "crisp" point p^* we will consider a "fuzzy point" $I : [0,1] \to [0,1]$ such that $I(p^*) = 1$ and $I(p) = 0$ for all p outside a small neighborhood of p^*. A standard choice would be something like $I := \mathbb{I}_{[p_-,p_+]}$, where $[p_-, p_+]$ is a short interval containing p and $\mathbb{I}_{[p_-,p_+]}$ is its indicator function, but we will want I to be continuous (it can, however, be arbitrarily close to $\mathbb{I}_{[p_-,p_+]}$).

The strategy for Skeptic ensuring (2) can be modified as follows. Let $\epsilon \in (0, 0.5]$ be again a small number. Now we consider the strategy $S_n(p) = S^{\epsilon,I}_n(p) := \epsilon I(p)\mathcal{K}_{n-1}$. Since

$$\mathcal{K}_n = \prod_{i=1}^{n}(1+\epsilon I(p_i)(y_i-p_i)),$$

on the paths where $\mathcal{K}_n$ is bounded we have

$$\prod_{i=1}^{n}(1+\epsilon I(p_i)(y_i-p_i)) \le C,$$

$$\sum_{i=1}^{n}\ln(1+\epsilon I(p_i)(y_i-p_i)) \le \ln C,$$

$$\epsilon\sum_{i=1}^{n} I(p_i)(y_i-p_i) - \epsilon^2\sum_{i=1}^{n} I^2(p_i)(y_i-p_i)^2 \le \ln C,$$

$$\epsilon\sum_{i=1}^{n} I(p_i)(y_i-p_i) - \epsilon^2\sum_{i=1}^{n} I(p_i) \le \ln C$$

(the last step involves replacing $I^2(p_i)$ with $I(p_i)$; the loss of precision is not great if I is close to $\mathbb{I}_{[p_-,p_+]}$),

$$\epsilon\sum_{i=1}^{n} I(p_i)(y_i-p_i) \le \ln C + \epsilon^2\sum_{i=1}^{n} I(p_i),$$

$$\frac{\sum_{i=1}^n I(p_i)(y_i - p_i)}{\sum_{i=1}^n I(p_i)} \le \frac{\ln C}{\epsilon \sum_{i=1}^n I(p_i)} + \epsilon.$$

The last inequality shows that the mean of y_i for p_i close to p^* is close to p^* provided we have observed sufficiently many such p_i; its interpretation is especially simple when I is close to $\mathbb{I}_{[p_-,p_+]}$.

In general, we may consider a mixture of $S_n^{\epsilon,I}(p)$ and $S_n^{-\epsilon,I}(p)$ for different values of ϵ and for different I covering all $p^* \in [0,1]$. If we make sure that the mixture is continuous (which is always the case for continuous I and finitely many ϵ and I), Theorem 1 provides us with forecasts that are unbiased in the small.

4.3 USING THE OBJECTS

Unbiasedness, even in the small, is only a necessary but far from sufficient condition for good forecasts: for example, a forecaster who ignores the objects x_n can be perfectly calibrated, no matter how much useful information x_n contain. (Cf. the discussion of resolution in [2]; we prefer not to use the term "resolution", which is too closely connected with the very special way of probability forecasting based on sorting and labeling.) It is easy to make the algorithm of the previous subsection take the objects into account: we can allow the test functions I to depend not only on p but also on the current object x_n; $S_n(p)$ then becomes a mixture of

$$S_n^{\epsilon,I}(p) := \epsilon I(p, x_n) \prod_{i=1}^{n-1} (1 + \epsilon I(p_i, x_i)(y_i - p_i))$$

and $S_n^{-\epsilon,I}(p)$ (defined analogously) over ϵ and I.

4.4 RELATION TO A STANDARD COUNTER-EXAMPLE

Suppose, for simplicity, that objects are absent ($|\mathbf{X}| = 1$). The standard construction from Dawid [1] showing that no forecasting strategy produces forecasts p_n that are unbiased in the small for all sequences is as follows. Define an infinite sequence $y_1, y_2, \dots$ recursively by

$$y_n := \begin{cases} 1 & \text{if } p_n < 0.5 \\ 0 & \text{otherwise,} \end{cases}$$

where p_n is the forecast produced by the forecasting strategy after seeing $y_1, \dots, y_{n-1}$. For the forecasts $p_n < 0.5$ we always have $y_n = 1$ and for the forecasts $p_n \ge 0.5$ we always have $y_n = 0$; obviously, we do not have unbiasedness in the small.

Let us see what Dawid's construction gives when applied to the defensive forecasting strategy constructed from the mixture of $S_n^{\epsilon,I}(p)$ and $S_n^{-\epsilon,I}(p)$, as described above, over different ϵ and different I; we will assume not only that the test functions I cover all $[0,1]$ but also that each point $p \in [0,1]$ is covered by arbitrarily narrow (concentrated in a small neighborhood of p) test functions. It is clear that we will inevitably have $p_n \to 0.5$ if p_n are produced by the defensive forecasting strategy and y_n are produced by Dawid's construction. On the other hand, since all test functions I are continuous and so cannot sharply distinguish between the cases $p_n < 0.5$ and $p_n \ge 0.5$, we do not have any contradiction: neither the test functions nor any observer who can only measure the p_n with a finite precision can detect the lack of unbiasedness in the small.

In this paper we are only interested in unbiasedness in the small when the test functions I are required to be continuous. Dawid's construction shows that unbiasedness in the small is impossible to achieve if I are allowed to be indicator functions of intervals (such as $[0, 0.5)$ and $[0.5, 1]$). To achieve unbiasedness in the small in this stronger sense, randomization appears necessary (see, e.g., [18]). It is interesting that already a little bit of randomization suffices, as explained in [5].

5 SIMPLIFIED ALGORITHM

Let us assume first that objects are absent, $|\mathbf{X}| = 1$. It was observed empirically that the performance of defensive forecasting strategies with a fixed ϵ does not depend on ϵ much (provided it is not too large; e.g., in the above calculations we assumed $\epsilon \le 0.5$). This suggests letting $\epsilon \to 0$ (in particular, we will assume that $\epsilon \ll n^{-2}$). As the test functions I we will take Gaussian bells I_j with standard deviation $\sigma > 0$ located densely and uniformly in the interval $[0,1]$. Letting $\approx$ stand for approximate equality and using the shorthand $\sum_\pm f(\pm) := f(+) + f(-)$, we obtain:

$$S_n(p) = \sum_\pm \sum_j (\pm\epsilon) I_j(p) \prod_{i=1}^{n-1} (1 \pm \epsilon I_j(p_i)(y_i - p_i))$$

$$= \sum_\pm \sum_j (\pm\epsilon) I_j(p) \exp\left(\sum_{i=1}^{n-1} \ln(1 \pm \epsilon I_j(p_i)(y_i - p_i)) \right)$$

$$\approx \sum_\pm \sum_j (\pm\epsilon) I_j(p) \exp\left(\pm\epsilon \sum_{i=1}^{n-1} I_j(p_i)(y_i - p_i) \right)$$

$$\approx \sum_\pm \sum_j (\pm\epsilon) I_j(p) \left(1 \pm \epsilon \sum_{i=1}^{n-1} I_j(p_i)(y_i - p_i) \right)$$

$$= \sum_\pm \sum_j (\pm\epsilon) I_j(p) \left(\pm\epsilon \sum_{i=1}^{n-1} I_j(p_i)(y_i - p_i) \right)$$

$$\propto \sum_j I_j(p) \sum_{i=1}^{n-1} I_j(p_i)(y_i - p_i)$$
$$= \sum_{i=1}^{n-1} K(p, p_i)(y_i - p_i), \tag{8}$$

where $K(p, p_i)$ is the Mercer kernel

$$K(p, p_i) := \sum_j I_j(p) I_j(p_i).$$

This Mercer kernel can be approximated by

$$\int_0^1 \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(t-p)^2}{2\sigma^2}\right) \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(t-p_i)^2}{2\sigma^2}\right) dt$$

$$\propto \int_0^1 \exp\left(-\frac{(t-p)^2 + (t-p_i)^2}{2\sigma^2}\right) dt$$
$$\approx \int_{-\infty}^{\infty} \exp\left(-\frac{(t-p)^2 + (t-p_i)^2}{2\sigma^2}\right) dt.$$

As a function of p, the last expression is proportional to the density of the sum of two Gaussian random variables of variance σ^2; therefore, it is proportional to

$$\exp\left(-\frac{(p-p_i)^2}{4\sigma^2}\right).$$

To get an idea of the properties of this forecasting strategy, which we call the *K29* strategy (or algorithm), we run it and the Laplace forecasting strategy ($p_n := (k+1)/(n+1)$, where k is the number of 1s observed so far) on a randomly generated bit sequence of length 1000 (with the probability of 1 equal to 0.5). A zero point p_n of S_n was found using the simple bisection procedure (see, e.g., [9], §§9.2–9.4, for more sophisticated methods): (a) start with the interval $[0, 1]$; (b) let p be the mid-point of the current interval; (c) if $S_n(p) > 0$, remove the left half of the current interval; otherwise, remove its right half; (d) go to (b). We did 10 iterations, after which the mid-point of the remaining interval was output as p_n. Notice that the values $S_n(0)$ and $S_n(1)$ did not have to be tested. Our program was written in MATLAB, Version 7, and the initial state of the random number generator was set to 0.

Figure 1 shows that the probabilities output by the K29 ($\sigma = 0.01$) and Laplace forecasting strategies are almost indistinguishable. To see that these two forecasting strategies can behave very differently, we complemented the 1000 bits generated as described above with 1000 0s followed by 1000 1s. The result is shown in Figure 2. The K29 strategy detects that the probability p of 1 changes after the 1000th round, and fairly quickly moves down. When the probability changes again after the 2000th round, K29 starts moving toward $p = 1$, but interestingly, hesitates around the line $p = 0.5$, as if expecting the process to reverse to the original probability of 1.

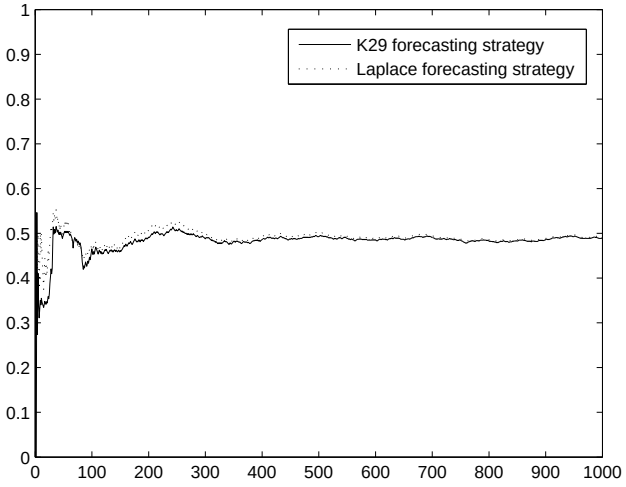


Figure 1: The First 1000 Probabilities Output by the K29 ($\sigma = 0.01$) and Laplace Forecasting Strategies on a Randomly Generated Bit Sequence

The Mercer kernel

$$K(p, p_i) = \exp\left(-\frac{(p-p_i)^2}{4\sigma^2}\right)$$

used in these experiments is known in machine learning as the Gaussian kernel (in the usual parameterization $4\sigma^2$ is replaced by $2\sigma^2$ or c); however, many other Mercer kernels also give reasonable results.

If we start from test functions I depending on the object, instead of (8) we will arrive at the expression

$$S_n(p) = \sum_{i=1}^{n-1} K((p, x_n), (p_i, x_i))(y_i - p_i), \tag{9}$$

where K is a Mercer kernel on the squared product $([0, 1] \times \mathbf{X})^2$. There are standard ways of constructing such Mercer kernels from Mercer kernels on $[0, 1]^2$ and $\mathbf{X}^2$ (see, e.g., the description of tensor products and direct sums in [13, 11]). For S_n to be continuous, we have to require that K be *forecast-continuous* in the following sense: for all $x \in \mathbf{X}$ and all $(p', x') \in [0, 1] \times \mathbf{X}$, $K((p, x), (p', x'))$ is continuous as a function of p. The overall procedure can be summarized as follows.

K29 Algorithm

Parameter: forecast-continuous Mercer kernel K on $([0, 1] \times \mathbf{X})^2$

FOR $n = 1, 2, \ldots$:
Read $x_n \in \mathbf{X}$.

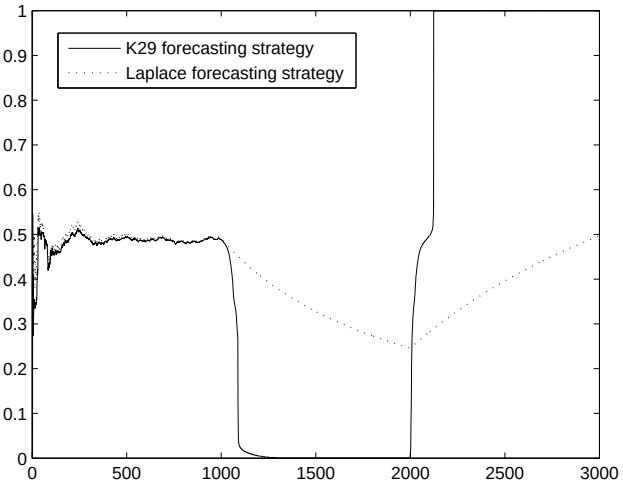


Figure 2: The Probabilities Output by the K29 ($\sigma = 0.01$) and Laplace Forecasting Strategies on a Randomly Generated Sequence of 1000 Bits Followed by 1000 0s and 1000 1s

Define $S_n(p)$ as per (9).
Output any root p of $S_n(p) = 0$ as p_n;
if there are no roots, $p_n := (1 + \text{sign}(S_n))/2$.
Read $y_n \in \{0, 1\}$.

Computer experiments reported in [16] show that the K29 algorithm performs well on a standard benchmark data set. For a theoretical discussion of the K29 algorithm, see [19] (Appendix) and [17].

6 RELATED WORK AND DIRECTIONS OF FURTHER RESEARCH

This paper's methods connect two areas that have been developing independently so far: probability forecasting and classical probability theory. It appears that, when properly developed, these methods can benefit both areas:

- the powerful machinery of classical probability theory can be used for probability forecasting;
- practical problems of probability forecasting may suggest new laws of probability.

Classical probability theory started from Bernoulli's weak law of large numbers (1713) and is the subject of countless monographs and textbooks. The original statements of most of its results were for independent random variables, but they were later extended to the martingale framework; the latter was reduced to its game-theoretic core in [12]. The proof of the strong law of large numbers used in this paper was extracted from Ville's [14] martingale proof of the law of the iterated logarithm (upper half).

The theory of probability forecasting was a topic of intensive research in meteorology in the 1960s and 1970s; this research is summarized in [2]. Machine learning is still mainly concerned with categorical prediction, but the situation appears to be changing. Probability forecasting using Bayesian networks is a mature field; the literature devoted to probability forecasting using decision trees and to calibrating other algorithms is also fairly rich. So far, however, the field of probability forecasting has been developing without any explicit connections with classical probability theory.

Defensive forecasting is indirectly related, in a sense dual, to prediction with expert advice (reviewed in [15], §4) and its special case, Bayesian prediction. In prediction with expert advice one starts with a given loss function and tries to make predictions that lead to a small loss as measured by that loss function. In defensive forecasting, one starts with a law of probability and then makes predictions such that this law of probability is satisfied. So the choice of the law of probability when designing the forecasting strategy plays a role analogous to the choice of the loss function in prediction with expert advice.

In prediction with expert advice one combines a pool of potentially promising forecasting strategies to obtain a forecasting strategy that performs not much worse than the best strategies in the pool. In defensive forecasting one combines strategies for Skeptic (such as the strategies corresponding to different test functions I and different $\pm\epsilon$ in §4) to obtain one strategy achieving an interesting goal (such as unbiasedness in the small); a strategy for Forecaster is then obtained using Theorem 1. The possibility of mixing strategies for Skeptic is as fundamental in defensive forecasting as the possibility of mixing strategies for Forecaster in prediction with expert advice.

This paper continues the work started by Foster and Vohra [4] and later developed in, e.g., [6, 10, 18] (the last paper replaces the von Mises–style framework of the previous papers with a martingale framework, as in this paper). The approach of this paper is similar to that of the recent paper [5], which also considers deterministic forecasting strategies and continuous test functions for unbiasedness in the small.

The main difference of this paper's approach from the bulk of work in learning theory is that we do not make any assumptions about Reality's strategy.

The following directions of further research appear to us most important:

- extending Theorem 1 to other forecasting protocols (such as multi-label classification) and designing efficient algorithms for finding the corresponding p_n;
- exploring forecasting strategies corresponding to: (a) Hoeffding's inequality, (b) the central limit theorem, (c) the law of the iterated logarithm (all we did in this paper was to slightly extend the strong law of large numbers and then use it for probability forecasting).

Acknowledgments

We are grateful to the participants of the PASCAL workshop "Notions of complexity: information-theoretic, computational and statistical approaches" (October 2004, EURANDOM) who commented on this work and to the anonymous referees for useful suggestions. This work was partially supported by BBSRC (grant 111/BIO14428), EPSRC (grant GR/R46670/01), MRC (grant S505/65), Royal Society, and, especially, the Superrobust Computation Project (Graduate School of Information Science and Technology, University of Tokyo).

References

[1] A. Philip Dawid. Self-calibrating priors do not exist: Comment. *Journal of the American Statistical Association*, 80:340–341, 1985. This is a contribution to the discussion in [8].

[2] A. Philip Dawid. Probability forecasting. In Samuel Kotz, Norman L. Johnson, and Campbell B. Read, editors, *Encyclopedia of Statistical Sciences*, volume 7, pages 210–218. Wiley, New York, 1986.

[3] Joseph L. Doob. *Stochastic Processes*. Wiley, New York, 1953.

[4] Dean P. Foster and Rakesh V. Vohra. Asymptotic calibration. *Biometrika*, 85:379–390, 1998.

[5] Sham M. Kakade and Dean P. Foster. Deterministic calibration and Nash equilibrium. In John Shawe-Taylor and Yoram Singer, editors, *Proceedings of the Seventeenth Annual Conference on Learning Theory*, volume 3120 of *Lecture Notes in Computer Science*, pages 33–48, Heidelberg, 2004. Springer.

[6] Ehud Lehrer. Any inspection is manipulable. *Econometrica*, 69:1333–1347, 2001.

[7] Allan H. Murphy and Edward S. Epstein. Verification of probabilistic predictions: a brief review. *Journal of Applied Meteorology*, 6:748–755, 1967.

[8] David Oakes. Self-calibrating priors do not exist (with discussion). *Journal of the American Statistical Association*, 80:339–342, 1985.

[9] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, second edition, 1992.

[10] Alvaro Sandroni, Rann Smorodinsky, and Rakesh V. Vohra. Calibration with many checking rules. *Mathematics of Operations Research*, 28:141–153, 2003.

[11] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[12] Glenn Shafer and Vladimir Vovk. *Probability and Finance: It's Only a Game!* Wiley, New York, 2001.

[13] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[14] Jean Ville. *Etude critique de la notion de collectif*. Gauthier-Villars, Paris, 1939.

[15] Vladimir Vovk. Competitive on-line statistics. *International Statistical Review*, 69:213–248, 2001.

[16] Vladimir Vovk. Defensive forecasting for a benchmark data set, The Game-Theoretic Probability and Finance project, `http://probabilityandfinance.com`, Working Paper #9, September 2004.

[17] Vladimir Vovk. Non-asymptotic calibration and resolution, The Game-Theoretic Probability and Finance project, `http://probabilityandfinance.com`, Working Paper #11, November 2004.

[18] Vladimir Vovk and Glenn Shafer. Good randomized sequential probability forecasting is always possible, The Game-Theoretic Probability and Finance project, `http://probabilityandfinance.com`, Working Paper #7, June 2003 (revised September 2004).

[19] Vladimir Vovk, Akimichi Takemura, and Glenn Shafer. Defensive forecasting, The Game-Theoretic Probability and Finance project, `http://probabilityandfinance.com`, Working Paper #8, September 2004. This is a fuller version of the current paper, with an appendix added.

Inadequacy of interval estimates corresponding to variational Bayesian approximations

Bo Wang
Department of Statistics
University of Glasgow
Glasgow G12 8QQ
Scotland, U.K.

D. M. Titterington
Department of Statistics
University of Glasgow
Glasgow G12 8QQ
Scotland, U.K.

Abstract

In this paper we investigate the properties of the covariance matrices associated with variational Bayesian approximations, based on data from mixture models, and compare them with the true covariance matrices, corresponding to Fisher information matrices. It is shown that the covariance matrices from the variational Bayes approximations are normally 'too small' compared with those for the maximum likelihood estimator, so that resulting interval estimates for the parameters will be unrealistically narrow, especially if the components of the mixture model are not well separated.

1 INTRODUCTION

A standard paradigm for learning about the parameters of latent variable models from data is that of maximum likelihood. However, maximum likelihood is well known for its tendency to overfit the data. On the other hand, the Bayesian framework averages over all possible settings of the model parameters. As a result Bayesian inference does not suffer from overfitting, and, moreover, prior knowledge can be incorporated naturally. Unfortunately, for most models of interest involving missing data a full Bayesian analysis requires the computation of the posterior distribution for a collection of unknown quantities, including parameters and latent variables, which often leads to intractable calculations because complicated multiple integrations are involved. The use of Markov chain Monte Carlo methods for numerical integration helps to side-step this problem, but this is clearly quite expensive, in terms of time and storage. Moreover MCMC algorithms can still exhibit conceptual and technical difficulties, for example in the assessment of the convergence of the chain to its stationary distribution.

Recently, a deterministic approximate approach to the intractable Bayesian learning problem, the variational Bayesian approximation, has been introduced in the machine learning community, and is widely recognised to be effective and promising in a variety of models, such as hidden Markov models (MacKay (1997)), graphical models (Attias (1999, 2000)), mixture models (Humphreys and Titterington (2000); Penny and Roberts (2000)), mixtures of factor analysers (Ghahramani and Beal (2000)) and state space models (Ghahramani and Beal (2001); Beal (2003)). A general formulation of the variational approach is described in Jordan (2004). The variational Bayes approach facilitates analytical calculation of approximate posterior distributions over the hidden variables, parameters and structures. They are computed via an iterative algorithm that is closely related to the Expectation-Maximisation (EM) algorithm and so its convergence is guaranteed. Empirically, variational Bayesian approximations have often been shown to perform well in earlier contributions, but it has also been noticed that this approach may underestimate the spread of the posterior distributions for some particular examples (Humphreys and Titterington (2000); Consonni and Marin (2004)), so its validity has still to be assessed properly: exact theoretical analysis of the quality of the method needs to be studied.

Some initial investigations have been implemented by the authors in Wang and Titterington (2003) and Wang and Titterington (2004b). It was shown theoretically that the iterative algorithm for obtaining the variational Bayes approximation for the parameters of Gaussian mixture models converges locally to the maximum likelihood estimator at the rate of $O(1/n)$ in the large sample limit. Later in Wang and Titterington (2004a) we proved local convergence of variational approximation algorithms for more general models, namely exponential family models with missing values, and showed that the variational posterior distribution for the parameters is asymptotically normal with the same mean but a different covariance ma-

trix compared with those for the maximum likelihood estimator.

Since the maximum likelihood estimators and posterior distributions are also asymptotically normal (see for instance Walker (1969), Chen (1985) and Ghosal et al. (1995)), an interesting problem is how these two limiting normal distributions can be compared. From the early results on local convergence of variational approximations, one can note that they have the same mean (i.e. the true value). However, their covariance matrices do not appear to be equal. In the context of Gaussian mixture models, in this paper we study the covariance matrices associated with variational Bayesian approximations, which dictate the performance of variational Bayes approximations for interval estimates, and compare them with the true covariance matrices, as given in terms of Fisher information matrices. We show that the covariance matrices from the variational Bayes approximation are normally 'too small' compared with those for the MLE, so that resulting interval estimates for the parameters will be too narrow, especially if the components of the mixture model are not well separated. Some numerical examples illustrate the theoretical analysis.

2 THE MIXTURE MODEL AND THE VARIATIONAL APPROXIMATION

We consider a model in which we have a mixture of m multivariate Gaussian densities $p_1, p_2, \ldots, p_m$ with mean vectors $\mu_1, \ldots, \mu_m$ and precision (inverse covariance) matrices $\Gamma_1, \ldots, \Gamma_m$, respectively. Thus the density of an observation is given by

$$p(y_i|\Theta) = \sum_{s=1}^{m} p_s(y_i|\Theta) p(s_i = s|\Theta), \tag{1}$$

where $y_i \in \mathbb{R}^d$ denotes the ith observed data vector, and s_i indicates the hidden component that generated it. The components are labelled by $s = 1, 2, \ldots, m$, and component s has mixing coefficient $\pi_s = p(s_i = s|\Theta)$, for any i and $s = 1, 2, \ldots, m-1$. Consequently $\pi_m \triangleq p(s_i = m|\Theta) = 1 - \sum_{s=1}^{m-1} \pi_s$. We write the parameters collectively as

$$\boldsymbol{\pi} = \begin{pmatrix} \pi_1 \\ \vdots \\ \pi_{m-1} \end{pmatrix}, \ \boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_m \end{pmatrix}, \ \boldsymbol{\Gamma} = \begin{pmatrix} \mathrm{vec}(\Gamma_1) \\ \vdots \\ \mathrm{vec}(\Gamma_m) \end{pmatrix},$$

and $\Theta = (\boldsymbol{\pi}', \boldsymbol{\mu}', \boldsymbol{\Gamma}')'$. Here $\mathrm{vec}(A)$ is defined as the stacked columns of A.

We use conjugate priors on the parameters Θ. The mixing coefficients $\boldsymbol{\pi}$ follow a symmetric Dirichlet distribution $\mathcal{D}(\lambda^0)$. The precisions are independently Wishart, with $\Gamma_s \sim \mathcal{W}(\nu^0, \Phi^0)$. The means conditioned on the precisions are independently Gaussian, with $\mu_s|\Gamma_s \sim \mathcal{N}(\rho^0, \beta^0\Gamma_s)$, where $\beta^0\Gamma_s$ is the inverse covariance matrix of the Gaussian distribution.

The joint density of Θ, S and Y is

$$p(\Theta, S, Y) = p(\boldsymbol{\pi}) \prod_{s=1}^{m} p(\mu_s|\Gamma_s) p(\Gamma_s) \prod_{i=1}^{n} \pi_{s_i} p_{s_i}(y_i).$$

In the variational Bayes approach, we use an approximating density $q(S, \Theta)$ for $p(S, \Theta|Y)$, which factorises as

$$q(S, \Theta) = q^{(S)}(S) q^{(\Theta)}(\Theta), \tag{2}$$

and such that the factors are chosen to maximise the negative *free energy*

$$\int \sum_{\{S\}} q(S, \Theta) \log \frac{p(\Theta, S, Y)}{q(S, \Theta)} d\Theta. \tag{3}$$

As a result of the form of $p(\Theta, S, Y)$, it follows immediately that the optimal $q^{(S)}(S)$ and $q^{(\Theta)}(\Theta)$ must factorise as

$$q^{(S)}(S) = \prod_{i=1}^{n} q_i^{(S)}(s_i),$$

$$q^{(\Theta)}(\Theta) = q(\boldsymbol{\pi}) \prod_{s=1}^{m} q(\mu_s|\Gamma_s) q(\Gamma_s).$$

As in Attias (1999, 2000), Ghahramani and Beal (2000), Humphreys and Titterington (2000) and Penny and Roberts (2000), the remaining details of the variational posteriors can be obtained by the following iterative procedure. In turn, we perform the following two stages.

(i) Optimise $q^{(\Theta)}(\Theta)$ for fixed $\{q_i^{(S)}(s_i), i = 1, \ldots, n\}$. Since conjugate priors are used, these variational posteriors are functionally identical to the priors, with different hyperparameter values: the mixing coefficients $\boldsymbol{\pi}$ are jointly Dirichlet, with $q(\boldsymbol{\pi}) = \mathcal{D}(\boldsymbol{\pi} : \lambda_1, \ldots, \lambda_m)$; the precisions are independently Wishart, with $q(\Gamma_s) = \mathcal{W}(\Gamma_s : \nu_s, \Phi_s)$; and the means conditioned on the precisions are independently Gaussian, with $q(\mu_s|\Gamma_s) = \mathcal{N}(\mu_s : \rho_s, \beta_s\Gamma_s)$. Here $\mathcal{D}(\boldsymbol{\pi} : \lambda_1, \ldots, \lambda_m)$, $\mathcal{W}(\Gamma_s : \nu_s, \Phi_s)$ and $\mathcal{N}(\mu_s : \rho_s, \beta_s\Gamma_s)$ denote the relevant density functions. The hyperparameters are updated as follows:

$$\lambda_s = \sum_{i=1}^{n} r_{is} + \lambda^0, \tag{4}$$

$$\rho_s = \Big(\sum_{i=1}^{n} r_{is} y_i + \beta^0 \rho^0 \Big) \Big/ \Big(\sum_{i=1}^{n} r_{is} + \beta_0 \Big), \tag{5}$$

$$\beta_s = \sum_{i=1}^{n} r_{is} + \beta^0, \quad \nu_s = \sum_{i=1}^{n} r_{is} + \nu^0, \tag{6}$$

and

$$\Phi_s = \Phi^0 + \sum_{i=1}^{n} r_{is}(y_i - \bar{\mu}_s)(y_i - \bar{\mu}_s)' + \Big[(\sum_{i=1}^{n} r_{is})\beta^0(\bar{\mu}_s - \rho^0)(\bar{\mu}_s - \rho^0)'\Big] \Big/ \Big(\sum_{i=1}^{n} r_{is} + \beta_0\Big), \tag{7}$$

where

$$r_{is} = q_i^{(S)}(s_i = s), \quad \bar{\mu}_s = \Big(\sum_{i=1}^{n} r_{is} y_i\Big) \Big/ \Big(\sum_{i=1}^{n} r_{is}\Big).$$

(ii) Optimise $\{q_i^{(S)}(s_i), s_i = 1, \ldots, m, i = 1, \ldots, n\}$ for fixed $q^{(\Theta)}(\Theta)$. For $s = 1, \ldots, m$, this results in

$$r_{is} = q_i^{(S)}(s_i = s) \propto \tilde{\pi}_s \tilde{\Gamma}_s^{1/2} e^{-(y_i - \rho_s)'\bar{\Gamma}_s(y_i - \rho_s)/2} \cdot e^{-d/(2\beta_s)} \triangleq \gamma_{is},$$

where

$$\tilde{\pi}_s = \exp\{\int q(\boldsymbol{\pi}) \log \pi_s d\boldsymbol{\pi}\},$$
$$\tilde{\Gamma}_s = \exp\{\int q(\Gamma_s) \log |\Gamma_s| d\Gamma_s\},$$
$$\bar{\Gamma}_s = \nu_s \Phi_s^{-1}.$$

If we let $\gamma_i = \sum_{s=1}^{m} \gamma_{is}$, $i = 1, \ldots, n$, then $r_{is} = \gamma_{is}/\gamma_i$.

This iterative procedure can be initialised by taking, for each i and s,

$$r_{is} \propto \lambda^0 (\nu^0 \Phi^0)^{1/2} e^{-(y_i - \rho^0)'\nu^0\Phi^0(y_i - \rho^0)/2} \cdot e^{-d/(2\beta^0)}.$$

3 THE CONVERGENCE OF VARIATIONAL BAYES APPROXIMATIONS AND ASSOCIATED COVARIANCE MATRICES

Suppose that the true value of the parameter Θ is Θ^*. At the kth iteration of the iterative procedure (i) (ii), we define the variational Bayesian estimates $\boldsymbol{\pi}^{(k)}$, $\boldsymbol{\mu}^{(k)}$ and $\boldsymbol{\Gamma}^{(k)}$ of the parameters $\boldsymbol{\pi}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Gamma}$ as their variational posterior means corresponding to the distributions $q(\boldsymbol{\pi})$, $q(\mu_s|\Gamma_s)$ and $q(\Gamma_s)$ at the current iteration, thus the iterative procedure (i) (ii) suggests the following algorithm: starting with some initial values $\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\mu}^{(0)}$ and $\boldsymbol{\Gamma}^{(0)}$, the variational Bayesian estimates are computed recursively by

$$\pi_s^{(k+1)} = M_1(\boldsymbol{\pi}^{(k)}, \boldsymbol{\mu}^{(k)}, \boldsymbol{\Gamma}^{(k)}), \tag{8a}$$
$$\mu_s^{(k+1)} = M_2(\boldsymbol{\pi}^{(k)}, \boldsymbol{\mu}^{(k)}, \boldsymbol{\Gamma}^{(k)}), \tag{8b}$$
$$\Gamma_s^{(k+1)} = M_3(\boldsymbol{\pi}^{(k)}, \boldsymbol{\mu}^{(k)}, \boldsymbol{\Gamma}^{(k)}), \tag{8c}$$

where the maps M_1, M_2 and M_3 represent the iterative procedure in (i) (ii).

In Wang and Titterington (2004b), the following convergence property of the variational Bayes estimates defined by (8) has been proved.

Lemma 1. *With probability 1 as n approaches infinity, $\boldsymbol{\pi}^{(k)}$, $\boldsymbol{\mu}^{(k)}$ and $\boldsymbol{\Gamma}^{(k)}$ converge locally to the true values $\boldsymbol{\pi}^*$, $\boldsymbol{\mu}^*$ and $\boldsymbol{\Gamma}^*$; that is, they converge to the true values whenever the starting values are sufficiently near to $\boldsymbol{\pi}^*$, $\boldsymbol{\mu}^*$ and $\boldsymbol{\Gamma}^*$.*

Remark 1. *For general mixture models, because the negative free energy* (3) *may be multi-modal, the variational Bayes algorithm may converge to different limits if different starting values (or hyperparameters) are chosen. Therefore only local convergence was proved.*

Denote by $\otimes$ the Kronecker product. By (4)-(7) and the convergence property given by Lemma 1, one can easily obtain that, as n tends to infinity, $n\mathrm{Cov}(\boldsymbol{\pi}) \to$

$$\begin{pmatrix} \pi_1^*(1-\pi_1^*) & & -\pi_s^*\pi_k^* \\ & \ddots & \\ -\pi_s^*\pi_k^* & & \pi_{m-1}^*(1-\pi_{m-1}^*) \end{pmatrix} \triangleq \Lambda,$$

$$n\mathrm{Cov}(\Gamma_s) = 2n\nu_s^{(k)}(\Phi_s^{(k)})^{-1} \otimes (\Phi_s^{(k)})^{-1} = 2\nu_s^{(k)}(\Phi_s^{(k)} \otimes \Phi_s^{(k)})^{-1} \to 2\pi_s^{*-1}(\Gamma_s^* \otimes \Gamma_s^*),$$

$$\mathbb{E}(\mu_s) = \int \mu_s q^{(k)}(\mu_s) d\mu_s = \int \mu_s q^{(k)}(\mu_s|\Gamma_s) q^{(k)}(\Gamma_s) d\Gamma_s d\mu_s = \rho_s^{(k)},$$

and it follows that

$$n\mathrm{Cov}(\mu_s) = n\int (\mu_s - \rho_s^{(k)})(\mu_s - \rho_s^{(k)})' q^{(k)}(\mu_s) d\mu_s$$
$$= n\int (\mu_s - \rho_s^{(k)})(\mu_s - \rho_s^{(k)})' q^{(k)}(\mu_s|\Gamma_s) q^{(k)}(\Gamma_s) d\Gamma_s d\mu_s$$
$$= n\int (\beta_s^{(k)}\Gamma_s^{(k)})^{-1} q^{(k)}(\Gamma_s) d\Gamma_s$$
$$= n(\beta_s^{(k)})^{-1}(\nu_s^{(k)} - m - 1)^{-1}\Phi_s^{(k)} \to \pi_s^{*-1}\Gamma_s^{*-1}.$$

Moreover, letting μ_s^j and $\Gamma_s^{t\tau}$ denote any elements of μ_s and Γ_s, respectively, we have

$$n\mathrm{Cov}(\mu_s^j, \Gamma_s^{t\tau}) = n\int [\mu_s^j - \mathbb{E}(\mu_s^j)][\Gamma_s^{t\tau} - \mathbb{E}(\Gamma_s^{t\tau})] \cdot q^{(k)}(\mu_s) q^{(k)}(\Gamma_s) d\mu_s^j d\Gamma_s^{t\tau}$$
$$= n\int [\mu_s^j - \rho_s^{(k),j}][\Gamma_s^{t\tau} - \mathbb{E}(\Gamma_s^{t\tau})] \cdot q^{(k)}(\mu_s|\Gamma_s)(q^{(k)}(\Gamma_s))^2 d\mu_s^j d\Gamma_s^{t\tau} = 0,$$

and the other covariances between $\boldsymbol{\pi}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Gamma}$ are zero, by assumption (2).

Define

$$\Omega = \mathrm{diag}(\pi_s^{*-1}\Gamma_s^{*-1}), \quad \Sigma = \mathrm{diag}(2\pi_s^{*-1}(\Gamma_s^* \otimes \Gamma_s^*)).$$

Then the covariance matrix of Θ associated with the variational posterior distributions is such that

$$n\mathrm{Cov}(\Theta) \to \begin{pmatrix} \Lambda & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Omega & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Sigma \end{pmatrix} \triangleq \Psi. \tag{9}$$

4 COMPARISON OF VARIATIONAL COVARIANCE MATRICES WITH FISHER INFORMATION MATRICES

In this section we first give an explicit expression for the Fisher information matrix associated with our mixture model, and then compare it with the covariance matrix associated with variational Bayes approximations, which is crucial for the performance of interval estimates based on variational Bayes approximations.

In the sequel, we denote by y any random vector distributed according to the probability density of the form (1). Thus the Fisher information matrix per observation is given by

$$I(\Theta) = \int_{\mathbb{R}^d} [\nabla \log p(y|\Theta)][\nabla \log p(y|\Theta)]' p(y|\Theta) dy. \tag{10}$$

The Fisher information matrix plays an important role in determining the asymptotic distribution of maximum likelihood estimators. Under quite mild conditions, Redner and Walker (1984) stated the following property of asymptotic normality for the maximum likelihood estimator for mixture models.

Theorem 1. *Let $\tilde{\Theta}^n$ be the strongly consistent MLE of the parameter Θ. Then $\sqrt{n}(\tilde{\Theta}^n - \Theta^*)$ is asymptotically normally distributed with mean zero and covariance matrix $I(\Theta^*)^{-1}$.*

Let $L(\Theta) = \log p(y|\Theta)$ and, for $s = 1, \ldots, m$, let

$$\alpha_s = \frac{p_s(y|\Theta)}{p(y|\Theta)}, \quad \delta_s = y - \mu_s,$$
$$\sigma_s = \mathrm{vec}[\Gamma_s^{-1} - (y - \mu_s)(y - \mu_s)'].$$

One should bear in mind the dependencies of α_s, δ_s and σ_s on Θ or its components, which are omitted for the sake of clarity.

After a straightforward calculation we obtain

$$\frac{\partial L}{\partial \pi_s} = \alpha_s - \alpha_m, \quad s = 1, \ldots, m-1,$$
$$\frac{\partial L}{\partial \mu_s} = \pi_s \alpha_s \Gamma_s \delta_s, \quad s = 1, \ldots, m,$$
$$\frac{\partial L}{\partial \mathrm{vec}(\Gamma_s)} = \frac{1}{2}\pi_s \alpha_s \sigma_s, \quad s = 1, \ldots, m.$$

Let

$$Q = \begin{pmatrix} \alpha_1 - \alpha_m \\ \vdots \\ \alpha_{m-1} - \alpha_m \\ \pi_1 \alpha_1 \Gamma_1 \delta_1 \\ \vdots \\ \pi_m \alpha_m \Gamma_m \delta_m \\ \frac{1}{2}\pi_1 \alpha_1 \sigma_1 \\ \vdots \\ \frac{1}{2}\pi_m \alpha_m \sigma_m \end{pmatrix}.$$

Then the Fisher information matrix (10) can be rewritten as

$$I(\Theta) = \int_{\mathbb{R}^d} QQ' p(y|\Theta) dy = \mathbb{E}[QQ']. \tag{11}$$

The following lemma is a corollary of Schwarz's inequality, which has been used in Peters and Walker (1978).

Lemma 2. *If $\eta_s \geq 0$ for $s = 1, \cdots, m$ and $\sum_{s=1}^m \eta_s = 1$, then*

$$|\sum_{s=1}^m \xi_s \eta_s|^2 \leq \sum_{s=1}^m \xi_s^2 \eta_s$$

for any $\{\xi_s\}_{s=1,\cdots,m}$.

Moreover, after a tedious calculation the following equalities can be verified.

Lemma 3. *At the true value Θ^*, we have that, for $s = 1, \ldots, m$,*

$$\mathbb{E}(\alpha_s) = 1, \quad \mathbb{E}(\alpha_s \delta_s) = 0, \tag{12a}$$
$$\mathbb{E}(\alpha_s \sigma_s) = 0, \quad \mathbb{E}(\alpha_s \delta_s \sigma_s') = 0, \tag{12b}$$
$$\mathbb{E}(\alpha_s \sigma_s \sigma_s') = 2(\Gamma_s^* \otimes \Gamma_s^*)^{-1}. \tag{12c}$$

Now we state the main result of this section.

Theorem 2. *If Ψ is defined as in (9), then the Fisher information matrix satisfies*

$$I(\Theta^*)^{-1} \geq \Psi, \tag{13}$$

by which it is meant that $I(\Theta^*)^{-1} - \Psi$ is nonnegative definite.

Proof. Obviously, Ψ is positive definite, and thus it is sufficient to show that

$$\Theta' I(\Theta^*)\Theta \le \Theta'\Psi^{-1}\Theta = \boldsymbol{u}'\Lambda^{-1}\boldsymbol{u}+\boldsymbol{v}'\Omega^{-1}\boldsymbol{v}+\boldsymbol{W}'\Sigma^{-1}\boldsymbol{W}$$

for any

$$\Theta = \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \\ \boldsymbol{W} \end{pmatrix} = \begin{pmatrix} u_1 \\ \vdots \\ u_{m-1} \\ v_1 \\ \vdots \\ v_m \\ \mathrm{vec}(W_1) \\ \vdots \\ \mathrm{vec}(W_m) \end{pmatrix},$$

where u_s, v_s and W_s are elements of the vector spaces $\mathbb{R}$, $\mathbb{R}^d$ and the set of all real, symmetric $d\times d$ matrices, respectively, for each s.

In fact, by (11) one has that

$$\begin{aligned}
&\Theta' I(\Theta^*)\Theta = \Theta'\mathbb{E}(QQ')\Theta \\
=&\mathbb{E}\Big\{ \sum_{s=1}^{m-1}(\alpha_s-\alpha_m)u_s + \sum_{s=1}^{m}\pi_s^*\alpha_s\delta_s'\Gamma_s^* v_s \\
&\qquad + \sum_{s=1}^{m}\frac{1}{2}\pi_s^*\alpha_s\sigma_s'\mathrm{vec}(W_s)\Big\}^2 \\
=&\mathbb{E}\Big\{ \sum_{s=1}^{m}\pi_s^*\alpha_s\Big[u_s\pi_s^{*-1} + \delta_s'\Gamma_s^* v_s + \frac{1}{2}\sigma_s'\mathrm{vec}(W_s)\Big]\Big\}^2,
\end{aligned}$$

where we have defined $u_m = -\sum_{s=1}^{m-1} u_s$.

Noting that $\sum_{s=1}^m \pi_s^*\alpha_s = 1$ and applying Lemma 2, we have

$$\begin{aligned}
&\Theta'\mathbb{E}(QQ')\Theta \\
\le&\mathbb{E}\Big\{ \sum_{s=1}^{m}\pi_s^*\alpha_s\Big[u_s\pi_s^{*-1} + \delta_s'\Gamma_s^* v_s + \frac{1}{2}\sigma_s'\mathrm{vec}(W_s)\Big]^2\Big\} \\
=&\sum_{s=1}^{m}\mathbb{E}\Big\{ u_s^2\pi_s^{*-1}\alpha_s + \pi_s^*\alpha_s[\delta_s'\Gamma_s^* v_s]^2 \\
&\qquad + \frac{1}{4}\pi_s^*\alpha_s[\sigma_s'\mathrm{vec}(W_s)]^2 + 2u_s\alpha_s\delta_s'\Gamma_s^* v_s \\
&\quad + u_s\alpha_s\sigma_s'\mathrm{vec}(W_s) + \pi_s^*\alpha_s\delta_s'\Gamma_s^* v_s\sigma_s'\mathrm{vec}(W_s)\Big\} \\
=&\sum_{s=1}^{m}\mathbb{E}\big\{u_s^2\pi_s^{*-1}\alpha_s\big\} + \sum_{s=1}^{m}\mathbb{E}\Big\{\pi_s^*\alpha_s[\delta_s'\Gamma_s^* v_s]^2\Big\} \\
&+ \sum_{s=1}^{m}\mathbb{E}\Big\{\frac{1}{4}\pi_s^*\alpha_s[\sigma_s'\mathrm{vec}(W_s)]^2\Big\} \\
\triangleq& I_1 + I_2 + I_3,
\end{aligned}$$

where the last equality holds since the cross terms average to zero, by (12).

Clearly, one has

$$I_1 = \sum_{s=1}^{m}\mathbb{E}\big\{\alpha_s u_s^2\pi_s^{*-1}\big\} = \sum_{s=1}^{m} u_s^2\pi_s^{*-1}.$$

Note that $u_m = -\sum_{s=1}^{m-1} u_s$ and

$$\Lambda^{-1} = \begin{pmatrix} \pi_1^{*-1}+\pi_m^{*-1} & & \pi_m^{*-1} \\ & \ddots & \\ \pi_m^{*-1} & & \pi_{m-1}^{*-1}+\pi_m^{*-1} \end{pmatrix},$$

from which it can be easily checked that $\boldsymbol{u}'\Lambda^{-1}\boldsymbol{u} = I_1$.

By (12),

$$I_2 = \sum_{s=1}^{m}\mathbb{E}\Big\{\pi_s^*\alpha_s[\delta_s'\Gamma_s^* v_s]^2\Big\} = \sum_{s=1}^{m} v_s'\pi_s^*\Gamma_s^* v_s = \boldsymbol{v}'\Omega^{-1}\boldsymbol{v}.$$

Finally,

$$\begin{aligned}
I_3 &= \sum_{s=1}^{m}\mathbb{E}\Big\{\frac{1}{4}\pi_s^*\alpha_s\big[\sigma_s'\mathrm{vec}(W_s)\big]^2\Big\} \\
&= \sum_{s=1}^{m}\mathbb{E}\Big\{\frac{1}{4}\pi_s^*\alpha_s\Big[\mathrm{tr}\{[\Gamma_s^{*-1} - (y-\mu_s^*)(y-\mu_s^*)']W_s\}\Big]^2\Big\} \\
&= \sum_{s=1}^{m}\frac{1}{4}\pi_s^* \\
&\cdot \mathbb{E}\Big\{\alpha_s\Big[(\mathrm{tr}\{\Gamma_s^{*-1}W_s\})^2 + (\mathrm{tr}\{(y-\mu_s^*)(y-\mu_s^*)'W_s\})^2 \\
&\qquad - 2\mathrm{tr}\{\Gamma_s^{*-1}W_s\}\mathrm{tr}\{(y-\mu_s^*)(y-\mu_s^*)'W_s\}\Big]\Big\} \\
&= \sum_{s=1}^{m}\frac{1}{4}\pi_s^*\Big\{\mathbb{E}\Big[\alpha_s(\mathrm{tr}\{(y-\mu_s^*)(y-\mu_s^*)'W_s\})^2\Big] \\
&\qquad - (\mathrm{tr}\{\Gamma_s^{*-1}W_s\})^2\Big\}.
\end{aligned}$$

By expanding the matrices into expressions involving their components and noting (12c), one can check that

$$\begin{aligned}
&\mathbb{E}\Big[\alpha_s(\mathrm{tr}\{(y-\mu_s^*)(y-\mu_s^*)'W_s\})^2\Big] \\
=&2\mathrm{tr}\big\{(W_s\Gamma_s^{*-1})^2\big\} + (\mathrm{tr}\{\Gamma_s^{*-1}W_s\})^2.
\end{aligned}$$

Therefore,

$$\begin{aligned}
I_3 &= \sum_{s=1}^{m}\frac{1}{2}\pi_s^*\mathrm{tr}\big\{W_s\Gamma_s^{*-2}W_s\big\} \\
&= \sum_{s=1}^{m}\frac{1}{2}\pi_s^*[\mathrm{vec}(W_s)]'(\Gamma_s^*\otimes\Gamma_s^*)^{-1}\mathrm{vec}(W_s) \\
&= \boldsymbol{W}'\Sigma^{-1}\boldsymbol{W}.
\end{aligned}$$

The proof is complete. □

Table 1: The Fisher information (FI) matrices and the inverse of the variational covariance (IVC) matrices corresponding to Figure 1. Each cell contains a 2×2 matrix.

	(1)	(2)	(3)	(4)
FI	5.83 2.50	2.47 1.29	6.08 3.77	0.00 0.00
	2.50 5.83	1.29 0.91	3.77 5.50	0.00 11.11
IVC	5.83 2.50	5.83 3.33	6.67 3.33	4.50 2.50
	2.50 5.83	3.33 6.67	3.33 5.83	2.50 12.50

By Lemma 2 the equality in (13) holds if and only if the mixture model (1) has only one component. In other cases, there must exist overlapping. If the components are well separated or have smaller overlaps, the mixture distribution can be regarded approximately as multinomial. In this case, for a given observation y_i, there exists one $p_s(y_i)$ which is far larger than the others, and therefore the inverse of Fisher information matrix is close to the covariance matrix of the variational posterior distribution. Theorem 2 shows that if overlapping exists between the components of a mixture model then the covariance matrix from the variational Bayes approximation is 'too small' compared with that for the MLE, so that resulting interval estimates for the parameters will be too narrow.

5 NUMERICAL EXPERIMENTS

In this section we demonstrate our results with some simple mixtures of normal densities.

First we consider mixtures of three known univariate normal densities $p_1(\cdot)$, $p_2(\cdot)$ and $p_3(\cdot)$ with means μ_1, μ_2 and μ_3; all have unit variance. The mixing coefficients are π_1, π_2 and $1 - \pi_1 - \pi_2$, respectively. For different values of these parameters, we compute the corresponding Fisher information matrices and the covariance matrices of the variational posteriors. The mixture densities of some typical cases are plotted in Figure 1, and the corresponding Fisher information matrices and the inverses of variational covariance matrices are described in Table 1. Obviously, if the components in the mixture models are widely separated, these two matrices are very similar, whereas, if the components are nearly identical, they are very different. The latter behaviour is reflected particularly by the case (4) in Figure 1, where $p_1(\cdot)$ and $p_1(\cdot)$ are completely identical.

Next we consider a more general mixture model of two unknown normal densities. Their means, precisions and mixing coefficients are μ_1, Γ_1, π and μ_2, Γ_2, $1-\pi$,

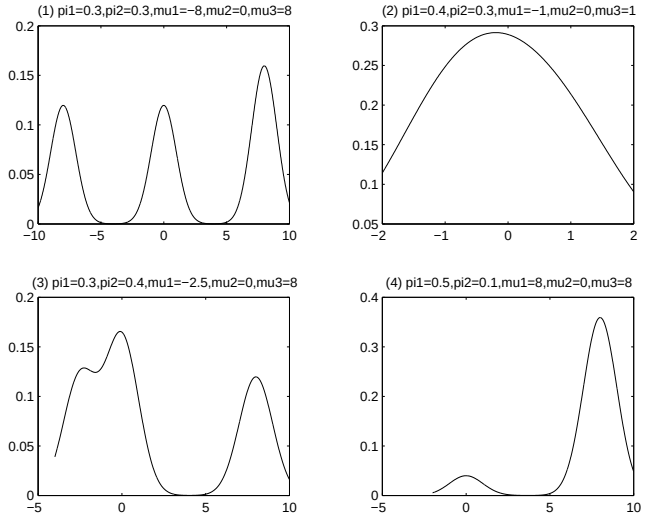


Figure 1: Some typical mixture densities based on different values of the parameters.

respectively. We compute the Fisher information matrices and the covariance matrices of the variational posteriors by using two sets of values of the parameters, namely, $(\pi, \mu_1, \Gamma_1, \mu_2, \Gamma_2) = (0.1, 1, 1, 0, 1)$ and $(0.5, 6, 1, 0, 1)$. There is large overlap between the two components for the first set of the parameters while they are well separated for the second. For the first set, the Fisher information matrix, denoted by I^*, is

$$\begin{pmatrix} 1.1542 & 0.1505 & 0.7456 & -0.0363 & -0.2612 \\ 0.1505 & 0.0259 & 0.0606 & -0.0134 & -0.0539 \\ 0.7456 & 0.0606 & 0.7723 & 0.0167 & 0.0774 \\ -0.0363 & -0.0134 & 0.0167 & 0.0152 & 0.0198 \\ -0.2612 & -0.0539 & 0.0774 & 0.0198 & 0.3646 \end{pmatrix}$$

and the inverse of the variational covariance matrix is

$$\Psi^{-1} = \mathrm{diag}(11.1111, 0.1000, 0.9000, 0.0500, 0.4500).$$

Evaluated at a couple of arbitrary vectors $\Theta = (0.8, 4, 3, 2, 1)'$ and $(1, 1, 1, 1, 1)'$, for illustrative purposes, $\Theta' I^* \Theta$ ($\Theta' \Psi^{-1} \Theta$) are equal to 14.0885 and 3.7435 (17.4611 and 12.6111), respectively. For the second set, the Fisher information matrix I^* is

$$\begin{pmatrix} 3.9834 & 0.0125 & 0.0125 & 0.0170 & -0.0170 \\ 0.0125 & 0.4905 & -0.0091 & -0.0133 & 0.0122 \\ 0.0125 & -0.0091 & 0.4905 & -0.0122 & 0.0133 \\ 0.0170 & -0.0133 & -0.0122 & 0.2308 & 0.0157 \\ -0.0170 & 0.0122 & 0.0133 & 0.0157 & 0.2308 \end{pmatrix}$$

and the inverse of the variational covariance matrix is

$$\Psi^{-1} = \mathrm{diag}(4.0000, 0.5000, 0.5000, 0.2500, 0.2500).$$

Evaluated at the same vectors Θ, $\Theta' I^* \Theta$ ($\Theta' \Psi^{-1} \Theta$) are equal to 15.7931 and 5.4889 (16.3100 and 5.5000), respectively.

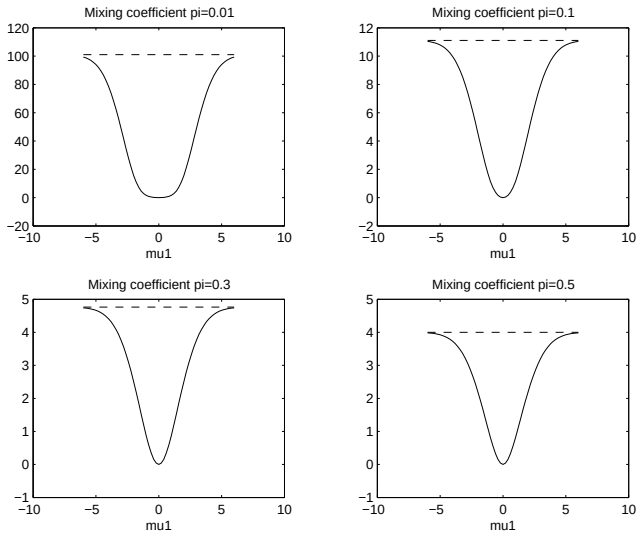


Figure 2: The inverses of the variances associated with the variational Bayes approximation and Fisher information for different mixing coefficients. The solid lines denote Fisher information and the dashed horizontal lines indicate the inverses of the variances for the variational Bayes approximation.

To clarify the dependence of the differences between the inverses of the variational covariance and Fisher information matrices on the overlaps between the components, now we use a mixture model of two known normal densities with means μ_1 and μ_2 with unit variance. The mixing coefficients are π and $1-\pi$, respectively. The parameter π is given a Beta prior distribution Beta$(1,1)$; i.e. $\pi \sim \text{Un}(0,1)$. To compare the variance associated with the variational Bayes approximate with Fisher information, we fix one component to have mean zero and compute the inverse of the variance and Fisher information with the other component having varying mean μ_1. The results are plotted in Figure 2 for different mixing coefficients π. The inverses of the variances associated with the variational Bayes approximation do not vary with the changes of μ_1, whereas the Fisher informations do. And the differences between them become larger as μ_1 is closer to zero, the mean of the first component.

We investigate the performance of interval estimates based on the variational approximation using two empirical experiments. We fix the mixing coefficient at $\pi^* = 0.65$ and one component to have mean zero and unit variance within a mixture model of two normal densities. Independent random samples, each of size $n = 50$, are selected from the mixture model with the mean of the other component equal to $\mu_2 = 3.0$ and 1.0, and with unit variance. For each sample we calculate the variational Bayesian estimate $\hat{\pi}$ as given by (8), the variational variance Λ, the maximum likelihood estimate $\tilde{\pi}$ and the Fisher information $I(\pi^*)$, and these are used to form approximate 95% confidence intervals given by $\hat{\pi} \pm 1.96\sqrt{\Lambda/n}$ and $\tilde{\pi} \pm 1.96/\sqrt{nI(\pi^*)}$. For $\mu_2 = 3.0$, a total of 100 samples are generated and the resulting 100 confidence intervals are computed. It turns out that 91 out of these 100 intervals do include the true value if the variational approximation is used, and 92 by the MLE method. Both proportions are close to the nominal confidence coefficient of 0.95. For $\mu_2 = 1.0$, the same number of confidence intervals are generated. Among these 100 intervals, only 68 of those based on the variational approximation include the true value, while this number is 92 from the MLE. In this case the resulting interval estimates are obviously too narrow.

Since the variational approaches provide good approximations for point estimates but poor approximations for interval estimates, a question of interest is whether or not the performance can be improved if we substitute the variational covariance matrix by the inverse of Fisher information for interval estimates. Theoretically, by this approach the resulting intervals would be very close to those obtained by MLE when the sample size is large, since the variational Bayes estimator converges to the maximum likelihood estimator. In order to verify this point, we use the same independent random samples as in the previous numerical examples to generate approximate 95% confidence interval given by $\hat{\pi} \pm 1.96/\sqrt{nI(\pi^*)}$. For $\mu_2 = 3.0$, 93 out of 100 intervals include the true value, whereas 96 intervals contain the true value if μ_2 is 1.0. It turns out that the approach of substituting the variational covariance matrix in the inverse of Fisher information does refine the interval estimates.

6 CONCLUSION

Exact theoretical analysis of the quality of variational Bayes approximations is an important issue. Having proved the properties of local convergence and asymptotic normality in Wang and Titterington (2003, 2004b,a), in this paper we examined the covariance matrices associated with variational Bayesian approximations and the resulting performance of variational Bayes approximations for interval estimates, by comparing them with the true covariances, given in terms of Fisher information matrices. It has been shown that the covariance matrices corresponding to the variational Bayes approximation are normally 'too small' compared with those for the MLE, so that resulting interval estimates for the parameters will be too narrow if the components of the mixture model are not well separated. Finally the theoretical analysis was reinforced by some numerical examples, which also sug-

gested an idea leading to the refinement of variational Bayes approximations for interval estimates by substituting the variational covariance by the 'usual' true covariance - the inverse of Fisher information. The arguments in the paper can be extended to non-Gaussian mixture models, such as mixtures of exponential family distributions, without any technical difficulty.

Acknowledgement

This work was supported by a grant from the UK Science and Engineering Research Council. This work was also supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

Attias, H. (1999). Inferring parameters and structure of latent variable models by variational Bayes. In Prade, H. and Laskey, K., editors, *Proc. 15th Conference on Uncertainty in Artificial Intelligence*, pages 21–30, Stockholm, Sweden. Morgan Kaufmann Publishers.

Attias, H. (2000). A variational Bayesian framework for graphical models. In Solla, S., Leen, T., and Muller, K.-R., editors, *Advances in Neural Information Processing Systems 12*, pages 209–215. MIT Press, Cambridge, MA.

Beal, M. J. (2003). *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, University College London.

Chen, C.-F. (1985). On asymptotic normality of limiting density functions with Bayesian implications. *J. R. Statist. Soc. B*, 47:540–546.

Consonni, G. and Marin, J.-M. (2004). A note on variational approximate bayesian inference for latent variable models. Preprint.

Ghahramani, Z. and Beal, M. J. (2000). Variational inference for Bayesian mixtures of factor analysers. In Solla, S., Leen, T., and Muller, K.-R., editors, *Advances in Neural Information Processing Systems 12*, pages 449–455. MIT Press, Cambridge, MA.

Ghahramani, Z. and Beal, M. J. (2001). Propagation algorithms for variational Bayesian learning. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 507–513. MIT Press, Cambridge, MA.

Ghosal, S., Ghosh, J. K., and Samanta, T. (1995). On convergence of posterior distributions. *The Annals of Statistics*, 23(6):2145–2152.

Humphreys, K. and Titterington, D. M. (2000). Approximate Bayesian inference for simple mixtures. In Bethlehem, J. G. and van der Heijden, P. G. M., editors, *COMPSTAT2000*, pages 331–336. Physica-Verlag, Heidelberg.

Jordan, M. I. (2004). Graphical models. *Statistical Science*, 19(1):140–155.

MacKay, D. J. C. (1997). Ensemble learning for hidden Markov models. Technical report, Cavendish Laboratory, University of Cambridge.

Penny, W. D. and Roberts, S. J. (2000). Variational Bayes for 1-dimensional mixture models. Technical Report PARG-2000-01, Oxford University.

Peters, B. C. and Walker, H. F. (1978). An iterative procedure for obtaining maximum-likelihood estimates of the parameters for a mixture of normal distributions. *SIAM J. Appl. Math.*, 35:362–378.

Redner, R. A. and Walker, H. F. (1984). Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26:195–239.

Walker, A. M. (1969). On the asymptotic behaviour of posterior distributions. *J. R. Statist. Soc. B*, 31:80–88.

Wang, B. and Titterington, D. M. (2003). Local convergence of variational Bayes estimators for mixing coefficients. Technical Report 03-4, University of Glasgow. http://www.stats.gla.ac.uk/Research/-TechRep2003/03-4.pdf.

Wang, B. and Titterington, D. M. (2004a). Convergence and asymptotic normality of variational Bayesian approximations for exponential family models with missing values. In Chickering, M. and Halpern, J., editors, *Proceedings of the twentieth conference on Uncertainty in Artificial Intelligence*, pages 577–584, Banff, Canada. AUAI Press.

Wang, B. and Titterington, D. M. (2004b). Convergence properties of a general algorithm for calculating variational Bayesian estimates for a normal mixture model. Technical Report 04-3, University of Glasgow. http://www.stats.gla.ac.uk/Research/-TechRep2004/04_3.pdf.

Nonlinear Dimensionality Reduction by Semidefinite Programming and Kernel Matrix Factorization

Kilian Q. Weinberger, Benjamin D. Packer, and Lawrence K. Saul*
Department of Computer and Information Science
University of Pennsylvania, Philadelphia, PA 19104-6389
{kilianw,lsaul,bpacker}@seas.upenn.edu

Abstract

We describe an algorithm for nonlinear dimensionality reduction based on semidefinite programming and kernel matrix factorization. The algorithm learns a kernel matrix for high dimensional data that lies on or near a low dimensional manifold. In earlier work, the kernel matrix was learned by maximizing the variance in feature space while preserving the distances and angles between nearest neighbors. In this paper, adapting recent ideas from semi-supervised learning on graphs, we show that the full kernel matrix can be very well approximated by a product of smaller matrices. Representing the kernel matrix in this way, we can reformulate the semidefinite program in terms of a much smaller submatrix of inner products between randomly chosen *landmarks*. The new framework leads to order-of-magnitude reductions in computation time and makes it possible to study much larger problems in manifold learning.

1 Introduction

A large family of graph-based algorithms has recently emerged for analyzing high dimensional data that lies or or near a low dimensional manifold [2, 5, 8, 13, 19, 21, 25]. These algorithms derive low dimensional embeddings from the top or bottom eigenvectors of specially constructed matrices. Either directly or indirectly, these matrices can be related to kernel matrices of inner products in a nonlinear feature space [9, 15, 22, 23]. These algorithms can thus be viewed as kernel methods with feature spaces that "unfold" the manifold from which the data was sampled.

*This work was supported by NSF Award 0238323.

In recent work [21, 22], we introduced Semidefinite Embedding (SDE), an algorithm for manifold learning based on semidefinite programming [20]. SDE learns a kernel matrix by maximizing the variance in feature space while preserving the distances and angles between nearest neighbors. It has several interesting properties: the main optimization is convex and guaranteed to preserve certain aspects of the local geometry; the method always yields a semipositive definite kernel matrix; the eigenspectrum of the kernel matrix provides an estimate of the underlying manifold's dimensionality; also, the method does not rely on estimating geodesic distances between faraway points on the manifold. This particular combination of advantages appears unique to SDE.

The main disadvantage of SDE, relative to other algorithms for manifold learning, is the time required to solve large problems in semidefinite programming. Earlier work in SDE was limited to data sets with $n \approx 2000$ examples, and problems of that size typically required several hours of computation on a mid-range desktop computer.

In this paper, we describe a new framework that has allowed us to reproduce our original results in a small fraction of this time, as well as to study much larger problems in manifold learning. We start by showing that for well-sampled manifolds, the entire kernel matrix can be very accurately reconstructed from a much smaller submatrix of inner products between randomly chosen *landmarks*. In particular, letting K denote the full $n \times n$ kernel matrix, we can write:

$$K \approx QLQ^T, \tag{1}$$

where L is the $m \times m$ submatrix of inner products between landmarks (with $m \ll n$) and Q is an $n \times m$ linear transformation derived from solving a sparse set of linear equations. The factorization in eq. (1) enables us to reformulate the semidefinite program in terms of the much smaller matrix L, yielding order-of-magnitude reductions in computation time.

The framework in this paper has several interesting connections to previous work in manifold learning and kernel methods. Landmark methods were originally developed to accelerate the multidimensional scaling procedure in Isomap [7]; they were subsequently applied to the fast embedding of sparse similarity graphs [11]. Intuitively, the methods in these papers are based on the idea of triangulation—that is, locating points in a low dimensional space based on their distances to a small set of landmarks. This idea can also be viewed as an application of the Nyström method [24, 12], which is a particular way of extrapolating a full kernel matrix from one of its sub-blocks. It is worth emphasizing that the use of landmarks in this paper is *not* based on this same intuition. SDE does not directly estimate geodesic distances between faraway inputs on the manifold, as in Isomap. As opposed to the Nyström method, our approach is better described as an adaptation of recent ideas for semi-supervised learning on graphs [1, 16, 18, 26, 27]. Our approach is somewhat novel in that we use these ideas not for transductive inference, but for computational savings in a purely *unsupervised* setting. To manage the many constraints that appear in our semidefinite programming problems, we have also adapted certain ideas from the large-scale training of support vector machines [6].

The paper is organized as follows. In section 2, we review our earlier work on manifold learning by semidefinite programming. In section 3, we investigate the kernel matrix factorization in eq. (1), deriving the linear transformation that reconstructs other examples from landmarks, and showing how it simplifies the semidefinite program for manifold learning. Section 4 gives experimental results on data sets of images and text. Finally, we conclude in section 5.

2 Semidefinite Embedding

We briefly review the algorithm for SDE; more details are given in previous work [21, 22]. As input, the algorithm takes high dimensional vectors $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$; as output, it produces low dimensional vectors $\{\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_n\}$. The inputs $\vec{x}_i \in \mathcal{R}^D$ are assumed to lie on or near a manifold that can be embedded in d dimensions, where typically $d \ll D$. The goal of the algorithm is to estimate the dimensionality d and to output a faithful embedding that reveals the structure of the manifold.

The main idea behind SDE has been aptly described as "maximum variance unfolding" [17]. The algorithm attempts to maximize the variance of its embedding, subject to the constraint that distances and angles between nearby inputs are preserved. The resulting transformation from inputs to outputs thus looks *locally* like a rotation plus translation—that is, it represents an isometry. To picture such a transformation from $D=3$ to $d=2$ dimensions, one can imagine a flag being unfurled by pulling on its four corners.

The first step of the algorithm is to compute the k-nearest neighbors of each input. A neighborhood-indicator matrix is defined as $\eta_{ij}=1$ if and only if the inputs $\vec{x}_i$ and $\vec{x}_j$ are k-nearest neighbors or if there exists another input of which both are k-nearest neighbors; otherwise $\eta_{ij}=0$. The constraints to preserve distances and angles between k-nearest neighbors can then be written as:

$$\|\vec{y}_i - \vec{y}_j\|^2 = \|\vec{x}_i - \vec{x}_j\|^2, \tag{2}$$

for all (i,j) such that $\eta_{ij}=1$. To eliminate a translational degree of freedom in the embedding, the outputs are also constrained to be centered on the origin:

$$\sum_i \vec{y}_i = \vec{0}. \tag{3}$$

Finally, the algorithm attempts to "unfold" the inputs by maximizing the variance

$$\mathrm{var}(\vec{y}) = \sum_i \|\vec{y}_i\|^2 \tag{4}$$

while preserving local distances and angles, as in eq. (2). Maximizing the variance of the embedding turns out to be a useful surrogate for minimizing its dimensionality (which is computationally less tractable).

The above optimization can be formulated as an instance of semidefinite programming [20]. Let $K_{ij} = \vec{y}_i \cdot \vec{y}_j$ denote the Gram (or kernel) matrix of the outputs. As shown in earlier work [21, 22], eqs. (2–4) can be written entirely in terms of the elements of this matrix. We can then learn the kernel matrix K by solving the following semidefinite program.

Maximize trace(K) subject to:
1) $K \succeq 0$**.**
2) $\Sigma_{ij} K_{ij} = 0$**.**
3) For all (i,j) **such that** $\eta_{ij}=1$**,**
$K_{ii} - 2K_{ij} + K_{jj} = \|\vec{x}_i - \vec{x}_j\|^2$**.**

As in kernel PCA [15], the embedding is derived from the eigenvalues and eigenvectors of the kernel matrix; in particular, the algorithm outputs $y_{\alpha i} = \sqrt{\lambda_\alpha} u_{\alpha i}$, where λ_α and u_α are the top d eigenvalues and eigenvectors. The dimensionality of the embedding, d, is suggested by the number of appreciably non-zero eigenvalues.

In sum, the algorithm has three steps: (i) computing k-nearest neighbors; (ii) computing the kernel matrix;

and (iii) computing its top eigenvectors. The computation time is typically dominated by the semidefinite program to learn the kernel matrix. In earlier work, this step limited us to problems with $n \approx 2000$ examples and $k \leq 5$ nearest neighbors; moreover, problems of this size typically required several hours of computation on a mid-range desktop computer.

3 Kernel Matrix Factorization

In practice, SDE scales poorly to large data sets because it must solve a semidefinite program over $n \times n$ matrices, where n is the number of examples. (Note that the computation time is prohibitive despite polynomial-time guarantees[1] of convergence for semidefinite programming.) In this section, we show that for well-sampled manifolds, the kernel matrix K can be approximately factored as the product of smaller matrices. We then use this representation to derive much simpler semidefinite programs for the optimization in the previous section.

3.1 Sketch of algorithm

We begin by sketching the basic argument behind the factorization in eq. (1). The argument has three steps. First, we derive a linear transformation for approximately reconstructing the entire data set of high dimensional inputs $\{\vec{x}_i\}_{i=1}^n$ from m randomly chosen inputs designated as *landmarks*. In particular, denoting these landmarks by $\{\vec{\mu}_\alpha\}_{\alpha=1}^m$, the reconstructed inputs $\{\hat{x}_i\}_{i=1}^n$ are given by the linear transformation:

$$\hat{x}_i = \sum_\alpha Q_{i\alpha} \vec{\mu}_\alpha. \tag{5}$$

The linear transformation Q is derived from a sparse weighted graph in which each node represents an input and the weights are used to propagate the positions of the m landmarks to the remaining $n-m$ nodes. The situation is analogous to semi-supervised learning on large graphs [1, 16, 18, 26, 27], where nodes represent labeled or unlabeled examples and transductive inferences are made by diffusion through the graph. In our setting, the landmarks correspond to labeled examples, the reconstructed inputs to unlabeled examples, and the vectors $\vec{\mu}_\alpha$ to the actual labels.

Next, we show that the *same linear transformation can be used to reconstruct the unfolded data set*—that is, after the mapping from inputs $\{\vec{x}_i\}_{i=1}^n$ to outputs $\{\vec{y}_i\}_{i=1}^n$. In particular, denoting the unfolded landmarks by $\{\vec{\ell}_\alpha\}_{\alpha=1}^m$ and the reconstructed outputs by $\{\hat{y}_i\}_{i=1}^n$, we argue that $\vec{y}_i \approx \hat{y}_i$, where:

$$\hat{y}_i = \sum_\alpha Q_{i\alpha} \vec{\ell}_\alpha. \tag{6}$$

The connection between eqs. (5–6) will follow from the particular construction of the weighted graph that yields the linear transformation Q. This weighted graph is derived by appealing to the symmetries of linear reconstruction coefficients; it is based on a similar intuition as the algorithm for manifold learning by locally linear embedding (LLE) [13, 14].

Finally, the kernel matrix factorization in eq. (1) follows if we make the approximation

$$K_{ij} = \vec{y}_i \cdot \vec{y}_j \approx \hat{y}_i \cdot \hat{y}_j. \tag{7}$$

In particular, substituting eq. (6) into eq. (7) gives the approximate factorization $K \approx QLQ^T$, where $L_{\alpha\beta} = \vec{\ell}_\alpha \cdot \vec{\ell}_\beta$ is the submatrix of inner products between (unfolded) landmark positions.

3.2 Reconstructing from landmarks

To derive the linear transformation Q in eqs. (5–6), we assume the high dimensional inputs $\{\vec{x}_i\}_{i=1}^n$ are well sampled from a low dimensional manifold. In the neighborhood of any point, this manifold can be locally approximated by a linear subspace. Thus, to a good approximation, we can hope to reconstruct each input by a weighted sum of its r-nearest neighbors for some small r. (The value of r is analogous but not necessarily equal to the value of k used to define neighborhoods in the previous section.) Reconstruction weights can be found by minimizing the error function:

$$\mathcal{E}(W) = \sum_i \left\| \vec{x}_i - \sum\nolimits_j W_{ij} \vec{x}_j \right\|^2, \tag{8}$$

subject to the constraint that $\sum_j W_{ij} = 1$ for all j, and where $W_{ij} = 0$ if $\vec{x}_j$ is not an r-nearest neighbor of $\vec{x}_i$. The sum constraint on the rows of W ensures that the reconstruction weights are invariant to the choice of the origin in the input space. A small regularizer for weight decay can also be added to this error function if it does not already have a unique global minimum.

Without loss of generality, we now identify the first m inputs $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_m\}$ as landmarks $\{\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_m\}$ and ask the following question: is it possible to reconstruct (at least approximately) the remaining inputs given just the landmarks $\vec{\mu}_\alpha$ and the weights W_{ij}? For sufficiently large m, a unique reconstruction can be obtained by minimizing eq. (8) with respect to $\{\vec{x}_i\}_{i>m}$.

[1] For the examples in this paper, we used the SDP solver CSDP v4.9 [4] with time complexity of $O(n^3+c^3)$ per iteration for sparse problems with $n \times n$ target matrices and c constraints. It seems, however, that large constant factors can also be associated with these complexity estimates.

To this end, we rewrite the reconstruction error as a function of the inputs, in the form:

$$\mathcal{E}(X) = \sum_{ij} \Phi_{ij}\, \vec{x}_i \cdot \vec{x}_j, \tag{9}$$

where $\Phi = (I_n - W)^T (I_n - W)$ and I_n is the $n \times n$ identity matrix. It is useful to partition the matrix Φ into blocks distinguishing the m landmarks from the other (unknown) inputs:

$$\Phi \;=\; \begin{pmatrix} \overbrace{\Phi^{\ell\ell}}^{m} & \overbrace{\Phi^{\ell u}}^{n-m} \\ \Phi^{u\ell} & \Phi^{uu} \end{pmatrix} \tag{10}$$

In terms of this matrix, the solution with minimum reconstruction error is given by the linear transformation in eq. (5), where:

$$Q = \begin{pmatrix} I_m \\ (\Phi^{uu})^{-1}\Phi^{ul} \end{pmatrix}. \tag{11}$$

An example of this minimum error reconstruction is shown in Fig. 1. The first two panels show $n = 10000$ inputs sampled from a Swiss roll and their approximate reconstructions from eq. (5) and eq. (11) using $r = 12$ nearest neighbors and $m = 40$ landmarks.

Intuitively, we can imagine the matrix Φ_{ij} in eq. (9) as defining a sparse weighted graph connecting nearby inputs. The linear transformation reconstructing inputs from landmarks is then analogous to the manner in which many semi-supervised algorithms on graphs propagate information from labeled to unlabeled examples.

To justify eq. (6), we now imagine that the data set has been unfolded in a way that preserves distances and angles between nearby inputs. As noted in previous work [13, 14], the weights W_{ij} that minimize the reconstruction error in eq. (8) are invariant to translations and rotations of each input and its r-nearest neighbors. Thus, roughly speaking, if the unfolding looks locally like a rotation plus translation, then the same weights W_{ij} that reconstruct the inputs $\vec{x}_i$ from their neighbors should also reconstruct the outputs $\vec{y}_i$ from theirs. This line of reasoning yields eq. (6). It also suggests that if we could somehow learn to faithfully embed just the landmarks in a lower dimensional space, the remainder of the inputs could be unfolded by a simple matrix multiplication.

3.3 Embedding the landmarks

It is straightforward to reformulate the semidefinite program (SDP) for the kernel matrix $K_{ij} = \vec{y}_i \cdot \vec{y}_j$ in section 2 in terms of the smaller matrix $L_{\alpha\beta} = \vec{\ell}_\alpha \cdot \vec{\ell}_\beta$. In particular, appealing to the factorization $K \approx QLQ^T$, we consider the following SDP:

Maximize trace(QLQ^T) subject to:
1) $L \succeq 0$.
2) $\Sigma_{ij}(QLQ^T)_{ij} = 0$.
3) For all (i,j) **such that** $\eta_{ij} = 1$,
$(QLQ^T)_{ii} - 2(QLQ^T)_{ij} + (QLQ^T)_{jj} \leq \|\vec{x}_i - \vec{x}_j\|^2$.

This optimization is nearly but not quite identical to the previous SDP up to the substitution $K \approx QLQ^T$. The only difference is that we have changed the equality constraints in eq. (2) to inequalities. The SDP in section 2 is guaranteed to be feasible since all the constraints are satisfied by taking $K_{ij} = \vec{x}_i \cdot \vec{x}_j$ (assuming the inputs are centered on the origin). Because the matrix factorization in eq. (1) is only approximate, however, here we must relax the distance constraints to preserve feasibility. Changing the equalities to inequalities is the simplest possible relaxation; the trivial solution $L_{\alpha\beta} = 0$ then provides a guarantee of feasibility. In practice, this relaxation does not appear to change the solutions of the SDP in a significant way; the variance maximization inherent to the objective function tends to saturate the pairwise distance constraints, even if they are not enforced as strict equalities.

To summarize, the overall procedure for unfolding the inputs $\vec{x}_i$ based on the kernel matrix factorization in eq. (1) is as follows: (i) compute reconstruction weights W_{ij} that minimize the error function in eq. (8); (ii) choose landmarks and compute the linear transformation Q in eq. (11); (iii) solve the SDP for the landmark kernel matrix L; (iv) derive a low dimensional embedding for the landmarks $\vec{\ell}_\alpha$ from the eigenvectors and eigenvalues of L; and (v) reconstruct the outputs $\vec{y}_i$ from eq. (6). The free parameters of the algorithm are the number of nearest neighbors r used to derive locally linear reconstructions, the number of nearest neighbors k used to generate distance constraints in the SDP, and the number of landmarks m (which also constrains the rank of the kernel matrix). In what follows, we will refer to this algorithm as landmark SDE, or simply ℓSDE.

ℓSDE can be much faster than SDE because its main optimization is performed over $m \times m$ matrices, where $m \ll n$. The computation time in semidefinite programming, however, depends not only on the matrix size, but also on the number of constraints. An apparent difficulty is that SDE and ℓSDE have the same number of constraints; moreover, the constraints in the latter are not sparse, so that a naive implementation of ℓSDE can actually be much slower than SDE. This difficulty is surmounted in practice by solving the semidefinite program for ℓSDE while only explicitly monitoring a small fraction of the original constraints. To start, we feed an initial subset of constraints to

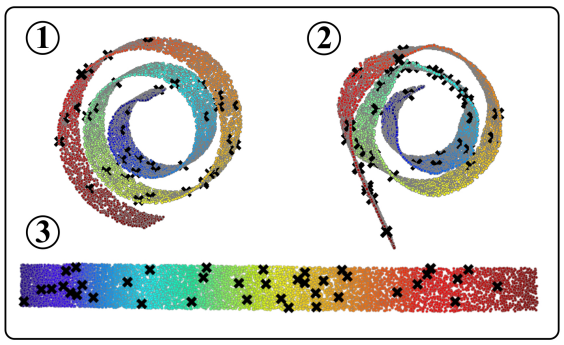


Figure 1: (1) $n = 10000$ inputs sampled from a Swiss roll; (2) linear reconstruction from $r = 12$ nearest neighbors and $m = 40$ landmarks (denoted by black x's); (3) embedding from ℓSDE, with distance and angle constraints to $k = 4$ nearest neighbors, computed in 16 minutes.

word	four nearest neighbors
one	two, three, four, six
may	won't, cannot, would, will
men	passengers, soldiers, officers, lawmakers
iraq	states, israel, china, noriega
drugs	computers, missiles, equipment, programs
january	july, october, august, march
germany	canada, africa, arabia, marks
recession	environment, yen, season, afternoon
california	minnesota, arizona, florida, georgia
republican	democratic, strong, conservative, phone
government	pentagon, airline, army, bush

Table 1: Selected words and their four nearest neighbors (in order of increasing distance) after nonlinear dimensionality reduction by ℓSDE. The $d = 5$ dimensional embedding of $D = 60000$ dimensional bigram distributions was computed by ℓSDE in 35 minutes (with $n = 2000$, $k = 4$, $r = 12$, and $m = 30$).

the SDP solver, consisting only of the semidefiniteness constraint, the centering constraint, and the distance constraints between landmarks and their nearest neighbors. If a solution is then found that violates some of the unmonitored constraints, these are added to the problem, which is solved again. The process is repeated until all the constraints are satisfied. Note that this incremental scheme is made possible by the relaxation of the distance constraints from equalities to inequalities. As in the large-scale training of support vector machines [6], it seems that many of the constraints in ℓSDE are redundant, and simple heuristics to prune these constraints can yield order-of-magnitude speedups. (Note, however, that the centering and semidefiniteness constraints in ℓSDE are always enforced.)

4 Experimental Results

Experiments were performed in MATLAB to evaluate the performance of ℓSDE on various data sets. The SDPs were solved with the CSDP (v4.9) optimization toolbox [4]. Of particular concern was the speed and accuracy of ℓSDE relative to earlier implementations of SDE.

The first data set, shown in the top left panel of Fig. 1, consisted of $n = 10000$ inputs sampled from a three dimensional "Swiss roll". The other panels of Fig. 1 show the input reconstruction from $m = 40$ landmarks and $r = 12$ nearest neighbors, as well as the embeddingobtained in ℓSDE by constraining distances and angles to $k = 4$ nearest neighbors. The computation took 16 minutes on a mid-range desktop computer. Table 2 shows that only 1205 out of 43182 constraints had to be explicitly enforced by the SDP solver to find a feasible solution. Interestingly, similarly faithful embeddings were obtained in shorter times using as few as $m = 10$ landmarks, though the input reconstructions in these cases were of considerably worse quality. Also worth mentioning is that adding low variance Gaussian noise to the inputs had no significant impact on the algorithm's performance.

The second data set was created from the $n = 2000$ most common words in the ARPA North American Business News corpus. Each of these words was represented by its discrete probability distribution over the $D = 60000$ words that could possibly follow it. The distributions were estimated from a maximum likelihood bigram model. The embedding of these high dimensional distributions was performed by ℓSDE (with $k = 4$, $r = 12$, and $m = 30$) in about 35 minutes; the variance of the embedding, as revealed by the eigenvalue spectrum of the landmark kernel matrix, was essentially confined to $d = 5$ dimensions. Table 1 shows a selection of words and their four nearest neighbors in the low dimensional embedding. Despite the massive dimensionality reduction from $D = 60000$ to $d = 5$, many semantically meaningful neighborhoods are seen to be preserved.

The third experiment was performed on $n = 400$ color images of a teapot viewed from different angles in the plane. Each vectorized image had a dimensionality of $D = 23028$, resulting from 3 bytes of color information for each of 76×101 pixels. In previous work [22] it was shown that SDE represents the angular mode of variability in this data set by an almost perfect circle. Fig. 2 compares embeddings from ℓSDE ($k = 4$, $r = 12$, $m = 20$) with normal SDE ($k = 4$) and LLE ($r = 12$). The eigenvalue spectrum of ℓSDE is very

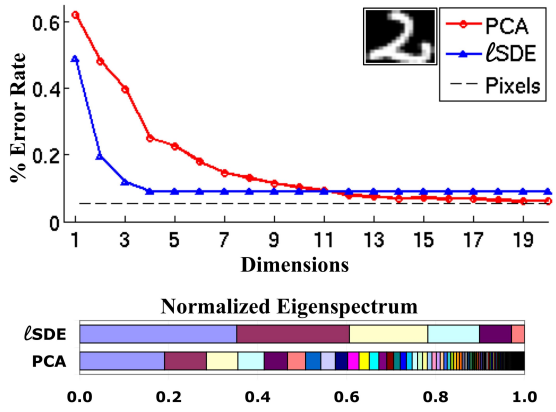


Figure 3: *Top:* Error rate of five-nearest-neighbors classification on the test set of USPS handwritten digits. The error rate is plotted against the dimensionality of embeddings from PCA and ℓSDE (with $k=4$, $r=12$, $m=10$). It can be seen that ℓSDE preserves the neighborhood structure of the digits fairly well with only a few dimensions. *Bottom:* Normalized eigenvalue spectra from ℓSDE and PCA. The latter reveals many more dimensions with appreciable variance.

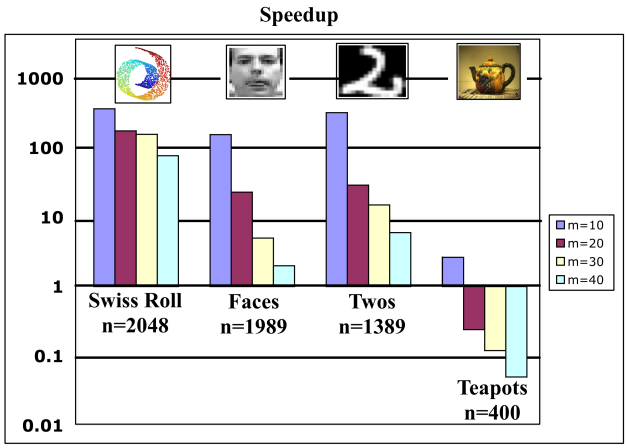


Figure 4: Relative speedup of ℓSDE versus SDE on data sets with different numbers of examples (n) and landmarks (m). Speedups of two orders of magnitude are observed on larger data sets. On small data sets, however, SDE can be faster than ℓSDE.

similar to that of SDE, revealing that the variance of the embedding is concentrated in two dimensions. The results from ℓSDE do not exactly reproduce the results from SDE on this data set, but the difference becomes smaller with increasing number of landmarks (at the expense of more computation time). Actually, as shown in Fig. 4, ℓSDE (which took 79 seconds) is slower than SDE on this particular data set. The increase in computation time has two simple explanations that seem peculiar to this data set. First, this data set is rather small, and ℓSDE incurs some overhead in its setup that is only negligible for large data sets. Second, this data set of images has a particular cyclic structure that is easily "broken" if the monitored constraints are not sampled evenly. Thus, this particular data set is not well-suited to the incremental scheme for adding unenforced constraints in ℓSDE; a large number of SDP reruns are required, resulting in a longer overall computation time than SDE. (See Table 2.)

The final experiment was performed on the entire data set of $n=9298$ USPS handwritten digits [10]. The inputs were 16×16 pixel grayscale images of the scanned digits. Table 2 shows that only 690 out of 61735 inequality constraints needed to be explicitly monitored by the SDP solver for ℓSDE to find a feasible solution. This made it possible to obtain an embedding in 40 minutes (with $k=4$, $r=12$, $m=10$), whereas earlier implementations of SDE could not handle problems of this size. To evaluate the embeddings from ℓSDE, we compared their nearest neighbor classification error rates to those of PCA. The top plot in Fig. 3 shows the classification error rate (using five nearest neighbors in the training images to classify test images) versus the dimensionality of the embeddings from ℓSDE and PCA. The error rate from ℓSDE drops very rapidly with dimensionality, nearly matching the error rate on the actual images with only $d=3$ dimensions. By contrast, PCA requires $d=12$ dimensions to overtake the performace of ℓSDE. The bar plot at the bottom of Fig. 3 shows the normalized eigenvalue spectra from both ℓSDE and PCA. From this plot, it is clear that ℓSDE concentrates the variance of its embedding in many fewer dimensions than PCA.

When does ℓSDE outperform SDE? Figure 4 shows the speedup of ℓSDE versus SDE on several data sets. Not surprisingly, the relative speedup grows in proportion with the size of the data set. Small data sets (with $n < 500$) can generally be unfolded faster by SDE, while larger data sets (with $500 < n < 2000$) can be unfolded up to 400 times faster by ℓSDE. For even larger data sets, only ℓSDE remains a viable option.

5 Conclusion

In this paper, we have developed a much faster algorithm for manifold learning by semidefinite programming. There are many aspects of the algorithm that we are still investigating, including the interplay between the number and placement of landmarks, the definition of local neighborhoods, and the quality of

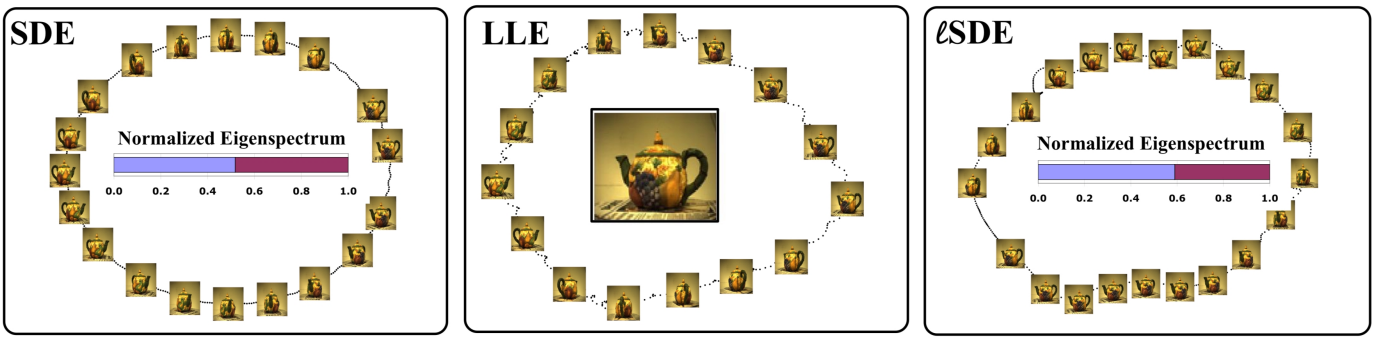


Figure 2: Comparison of embeddings from SDE, LLE and ℓSDE for $n = 400$ color images of a rotating teapot. The vectorized images had dimension $D = 23028$. LLE (with $r = 12$) and ℓSDE (with $k = 4$, $r = 12$, $m = 20$) yield similar but slightly more irregular results than SDE (with $k = 4$). The normalized eigenspectra in SDE and ℓSDE (i.e., the eigenspectra divided by the trace of their kernel matrices) reveal that the variances of their embeddings are concentrated in two dimensions; the eigenspectrum from LLE does not reveal this sort of information.

data set	n	m	constraints	monitored	time (secs)
teapots	400	20	1599	565	79
bigrams	2000	30	11170	1396	2103
USPS digits	9298	10	61735	690	2420
Swiss roll	10000	20	43182	1205	968

Table 2: Total number of constraints versus number of constraints explicitly monitored by the SDP solver for ℓSDE on several data sets. The numbers of inputs (n) and landmarks (m) are also shown, along with computation times. The speedup of ℓSDE is largely derived from omitting redundant constraints.

the resulting reconstructions and embeddings. Nevertheless, our initial results are promising and show that manifold learning by semidefinite programming can scale to much larger data sets than we originally imagined in earlier work [21, 22].

Beyond the practical applications of ℓSDE, the framework in this paper is interesting in the way it combines ideas from several different lines of recent work. ℓSDE is based on the same appeals to symmetry at the heart of LLE [13, 14] and SDE [21, 22]. The linear reconstructions that yield the factorization of the kernel matrix in eq. (1) are also reminiscent of semi-supervised algorithms for propagating labeled information through large graphs of unlabeled examples [1, 16, 18, 26, 27]. Finally, though based on a somewhat different intuition, the computational gains of ℓSDE are similar to those obtained by landmark methods for Isomap [7].

While we have applied ℓSDE (in minutes) to data sets with as many as $n = 10000$ examples, there exist many larger data sets for which the algorithm remains impractical. Further insights are therefore required. In related work, we have developed a simple out-of-sample extension for SDE, analogous to similar extensions for other spectral methods [3]. Algorithmic advances may also emerge from the dual formulation of "maximum variance unfolding" [17], which is related to the problem of computing fastest mixing Markov chains on graphs. We are hopeful that a combination of complementary approaches will lead to even faster and more powerful algorithms for manifold learning by semidefinite programming.

Acknowledgments

We are grateful to Ali Jadbabaie (University of Pennsylvania) for several discussions about semidefinite programming and to the anonymous reviewers for many useful comments.

References

[1] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In *Proceedings of the Seventeenth Annual Conference on Computational Learning Theory (COLT 2004)*, pages 624–638, Banff, Canada, 2004.

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[3] Y. Bengio, J-F. Paiement, and P. Vincent. Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press.

[4] B. Borchers. CSDP, a C library for semidefinite programming. *Optimization Methods and Software 11(1):613-623*, 1999.

[5] M. Brand. Charting a manifold. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 985–992, Cambridge, MA, 2003. MIT Press.

[6] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.

[7] V. de Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 721–728, Cambridge, MA, 2003. MIT Press.

[8] D. L. Donoho and C. E. Grimes. Hessian eigenmaps: locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Arts and Sciences*, 100:5591–5596, 2003.

[9] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, pages 369–376, Banff, Canada, 2004.

[10] J. J. Hull. A database for handwritten text recognition research. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 16(5):550–554, May 1994.

[11] J. C. Platt. Fast embedding of sparse similarity graphs. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press.

[12] J. C. Platt. FastMap, MetricMap, and landmark MDS are all nyström algorithms. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, Barbados, WI, January 2005.

[13] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[14] L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

[15] B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[16] A. J. Smola and R. Kondor. Kernels and regularization on graphs. In *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory and Kernel Workshop*, Washington D.C., 2003.

[17] J. Sun, S. Boyd, L. Xiao, and P. Diaconis. The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem. *SIAM Review*, submitted.

[18] M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[19] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[20] L. Vandenberghe and S. P. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, March 1996.

[21] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-04)*, volume 2, pages 988–995, Washington D.C., 2004.

[22] K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, pages 839–846, Banff, Canada, 2004.

[23] C. K. I. Williams. On a connection between kernel PCA and metric multidimensional scaling. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 675–681, Cambridge, MA, 2001. MIT Press.

[24] Christopher K. I. Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In T. Leen, T. Dietterich, and V. Tresp, editors, *Neural Information Processing Systems 13*, pages 682–688, Cambridge, MA, 2001. MIT Press.

[25] Z. Zhang and H. Zha. Principal manifolds and nonlinear dimensionality reduction by local tangent space alignment. *SIAM Journal of Scientific Computing*, in press.

[26] D. Zhou, O. Bousquet, T. N. Lai, J. Weston, and B. Schölkopf. Learning with local and global consistency. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 321–328, Cambridge, MA, 2004. MIT Press.

[27] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pages 912–919, Washington D.C., 2003.

An Expectation Maximization Algorithm for Inferring Offset-Normal Shape Distributions

Max Welling
School of Information and Computer Science
University of California Irvine
Irvine CA 92697-3425 USA
welling@ics.uci.edu

Abstract

The statistical theory of shape plays a prominent role in applications such as object recognition and medical imaging. An important parameterized family of probability densities defined on the locations of landmark-points is given by the offset-normal shape distributions introduced in [7]. In this paper we present an EM algorithm for learning the parameters of the offset-normal shape distribution from shape data. To improve model flexibility we also provide an EM algorithm to learn mixtures of offset-normal distributions. To deal with missing landmarks (e.g. due to occlusions), we extend the algorithm to train on incomplete data-sets. The algorithm is tested on a number of real-world data sets and on some artificially generated data. Experimentally, this seems to be the first algorithm for which estimation of the full covariance matrix causes no difficulties. In all experiments the estimated distribution provided an excellent approximation to the true offset-normal shape distribution.

1 INTRODUCTION

The statistical analysis of shape has important applications in fields as diverse as biology, anatomy, genetics, medicine, archeology, geology, geography, agriculture, image analysis, computer vision, pattern recognition and chemistry (see e.g. [9]). As an important example, we can represent an object (e.g. a face, skull, etc.) as a collection of landmarks at certain positions (in figure space). To compare objects it is then useful to discard differences in location, orientation and scale. (i.e. their pose). The remaining degrees of freedom are called the *shape* of an object. For a meaningful comparison of objects by their shape we need the tools of "statistical shape analysis". For instance, we may want to know whether two objects are *significantly* different (using a hypothesis test), or we may be interested in classifying or clustering objects by their shape. The statistical analysis of shape has a long history dating back to the late seventies [15, 10, 11, 12, 1, 2, 3, 4].

The work that we will present here is based on a more recent development in statistical shape analysis, namely the introduction of the *offset-normal* distribution [14, 7, 8, 9]. Offset-normal probability densities describe the distribution of shapes as represented by collections of landmark points in two dimensions. The assumption is that the landmarks in figure space are normally distributed. Pose is removed by mapping two landmarks to fixed positions (e.g. $(0,0)$ and $(1,0)$), while the remaining landmarks represent the shape information. Perhaps surprisingly, this distribution over the remaining landmarks can be expressed in analytic form [7]. However, a reliable method to infer the distribution parameters from shape data in the most general case (full covariance matrix), is not available. The fact that certain singular normal distributions map to the same offset-normal shape distribution has obstructed the formulation of estimation procedures for general covariance matrices.

In this paper we will derive EM update rules for unrestricted parameters of the offset-normal shape distribution, i.e. a mean vector and a full covariance matrix. As it turns out, both E- and M-step can be computed analytically, providing an efficient update scheme. In pattern recognition, it may happen that landmarks are occluded. To deal with this difficulty which is often encountered in practical problems we extend the EM procedure to learn from incomplete data. For cases where the data are not well approximated by an offset-normal shape distribution, we provide EM-learning rules for *mixtures* of offset-normal shape distributions. We conclude with experiments on some real world data-sets.

2 THE OFFSET-NORMAL SHAPE DISTRIBUTION

In order to be self contained, we explain and re-derive the offset-normal shape distribution in this section. Some results in later sections will follow a similar derivation.

Let an object in two dimensions be represented by the positions $\{x_i, y_i\}$ of p landmarks. Let $\mathbf{x}$ be distributed according to a $2p$ dimensional normal distribution, $\mathbf{x} \sim \mathcal{N}_{2p}[\boldsymbol{\nu}, \boldsymbol{\Omega}]$.

We will first remove translational content by applying the following transformation,

$$\mathbf{x} = [x_1, ..., x_p, y_1, ..., y_p]^T \rightarrow \mathbf{L}\mathbf{x} \tag{1}$$

with

$$\mathbf{L} = \begin{bmatrix} \mathbf{I} - \mathbf{1}\mathbf{e}_1^T + \mathbf{e}_1\mathbf{e}_1^T & \mathbf{0} \\ \mathbf{0} & \mathbf{I} - \mathbf{1}\mathbf{e}_1^T + \mathbf{e}_1\mathbf{e}_1^T \end{bmatrix} \tag{2}$$

and integrate out the first landmark. In this equation $\mathbf{I}$ and $\mathbf{0}$ are the $p \times p$ dimensional identity and zero matrices respectively, $\mathbf{1}$ is a $p \times 1$ dimensional vector of ones and $\mathbf{e}_1$ is the $p \times 1$ dimensional vector $[1, 0, ..., 0]^T$. This transformation shifts all landmarks, except the first one, by an amount x_1, y_1. Notice that if we had also shifted the first landmark, it would be fixed at the location $(0, 0)$, producing a singular probability distribution. Since the above transformation is linear, the coordinates $\mathbf{L}\mathbf{x}$ are also normally distributed with mean $\boldsymbol{\mu} = \mathbf{L}\boldsymbol{\nu}$ and covariance $\boldsymbol{\Sigma} = \mathbf{L}\boldsymbol{\Omega}\mathbf{L}^T$. Integrating out x_1, y_1 for a normal distribution is simply accomplished by deleting the corresponding entries in the mean and covariance. The remaining coordinates are denoted by $\mathbf{x}^* = [x_2^*, ..., x_p^*, y_2^*, ..., y_p^*]^T$ and have dimension $2p - 2$.

Next, we remove rotation and scale content by following a similar procedure. First, we transform $\mathbf{x}^*$ as follows,

$$\begin{aligned} u_2 &= x_2^*, \qquad u_i = \frac{(x_i^* x_2^* + y_i^* y_2)}{x_2^{*2} + y_2^{*2}} \quad i = 3, ..., p \\ v_2 &= y_2^*, \qquad v_i = \frac{(y_i^* x_2^* - x_i^* y_2)}{x_2^{*2} + y_2^{*2}} \quad i = 3, ..., p \end{aligned} \tag{3}$$

This transformation would have moved the second landmark to the location $(1, 0)$, not allowing any spread and generating a singular pdf. Therefore, we will leave the second landmark untouched, while treating all the other ones as if the second landmark were moved to the reference position $(1, 0)$. Finally, to remove information on orientation and scale we need to integrate out the second landmark, which we will do in the following.

We will simplify notation for the second landmark by writing $\mathbf{x}_2^* = \mathbf{h}$, while $\mathbf{u} = [u_3, ...u_p, v_3, ..., v_p]^T$. In the coordinates $\{\mathbf{h}, \mathbf{u}\}$ the pdf is given by,

$$P(\mathbf{h}, \mathbf{u}) = \frac{1}{(2\pi)^{p-1}\sqrt{\det \boldsymbol{\Sigma}}} \exp[-\frac{1}{2}G] |\det \mathbf{J}|, \tag{4}$$

with,

$$\begin{aligned} G &= (\mathbf{W}\mathbf{h} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{W}\mathbf{h} - \boldsymbol{\mu}), & (5) \\ \det \mathbf{J} &= (h_x^2 + h_y^2)^{p-2}, & (6) \end{aligned}$$

where $\mathbf{J}$ is the Jacobian of the transformation (3) and

$$\mathbf{W}^T = \begin{bmatrix} 1 & u_3 & \cdots & u_p & 0 & v_3 & \cdots & v_p \\ 0 & -v_3 & \cdots & -v_p & 1 & u_3 & \cdots & u_p \end{bmatrix}. \tag{7}$$

The integration over $\mathbf{h}$ is facilitated by rewriting G as,

$$G = (\mathbf{h} - \boldsymbol{\xi})^T \boldsymbol{\Gamma}^{-1} (\mathbf{h} - \boldsymbol{\xi}) + g \tag{8}$$

with

$$\begin{aligned} \boldsymbol{\Gamma}^{-1} &= \mathbf{W}^T \boldsymbol{\Sigma}^{-1} \mathbf{W} & (9) \\ \boldsymbol{\xi} &= \boldsymbol{\Gamma}\mathbf{W}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} & (10) \\ g &= \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} - \boldsymbol{\xi}^T \boldsymbol{\Gamma}^{-1} \boldsymbol{\xi} & (11) \end{aligned}$$

We can simplify (4) further by transforming to the eigenbasis of $\boldsymbol{\Gamma}$,

$$\begin{aligned} &\boldsymbol{\Gamma} = \mathbf{R}\mathbf{D}\mathbf{R}^T, \\ &\boldsymbol{\zeta} = \mathbf{R}^T \boldsymbol{\xi} \qquad \mathbf{z} = \mathbf{R}^T \mathbf{h}. \end{aligned} \tag{12}$$

Noticing that the determinant of the Jacobian is invariant with respect to rotations, this gives,

$$\begin{aligned} P(\mathbf{z}, \mathbf{u}) &= \frac{1}{(2\pi)^{p-2}} \sqrt{\frac{\det \boldsymbol{\Gamma}}{\det \boldsymbol{\Sigma}}} e^{-\frac{1}{2}g} \times \\ &\times \mathcal{N}_{z_x}[\zeta_x, \sigma_x] \, \mathcal{N}_{z_y}[\zeta_y, \sigma_y] \, (z_x^2 + z_y^2)^{p-2} \end{aligned} \tag{13}$$

where

$$\sigma_x = \sqrt{D_{xx}} \qquad \sigma_y = \sqrt{D_{yy}} \tag{14}$$

Finally, we use the binomial expansion to rewrite the Jacobian as,

$$(z_x^2 + z_y^2)^{p-2} = \sum_{i=0}^{p-2} \begin{pmatrix} p-2 \\ i \end{pmatrix} z_x^{2i} \, z_y^{2p-4-2i}. \tag{15}$$

We are now ready to perform the integrations over $\mathbf{h}$, required for the definition of the offset-normal shape distribution,

$$\begin{aligned} &P_{\mathcal{S}}(\mathbf{u}) = \int d\mathbf{h} \, p(\mathbf{h}, \mathbf{u}) = \int d\mathbf{z} \, p(\mathbf{z}, \mathbf{u}) = \\ &\frac{1}{(2\pi)^{p-2}} \sqrt{\frac{\det \boldsymbol{\Gamma}}{\det \boldsymbol{\Sigma}}} e^{-\frac{1}{2}g} \times \\ &\times \sum_{i=0}^{p-2} \begin{pmatrix} p-2 \\ i \end{pmatrix} \mathbf{E}[z_x^{2i} | \zeta_x, \sigma_x] \, \mathbf{E}[z_y^{2p-4-2i} | \zeta_y, \sigma_y] \end{aligned} \tag{16}$$

where,

$$\mathbf{E}[z^k | \mu, \sigma] = \left(\frac{\sqrt{2}\sigma}{2i} \right)^k H_k(\frac{i\mu}{\sqrt{2}\sigma}), \tag{17}$$

denotes a Gaussian expectation and H_k denotes the Hermite polynomial of order k. Equation (17) is the offset-normal shape distribution [7], which is invariant with respect to translations, rotations and scalings of the data. It is

expressed in terms of the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ which are not invariant with respect to orientation and scale changes (the translations were taken out in going from $\boldsymbol{\nu} \rightarrow \boldsymbol{\mu}, \boldsymbol{\Omega} \rightarrow \boldsymbol{\Sigma}$). It follows that the parameter set must be redundant, i.e. orientation and scale transformations of the parameters map to the same offset-normal shape distribution. Technically, this implies that the offset-normal shape distribution is described by an equivalence class of parameters. Therefore, when we mention in the rest of this paper that some random variable is distributed according to an offset-normal shape distribution with parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we refer to the equivalence class of all $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ that map to the same offset-normal shape distribution. Sometimes it will be useful to remove this ambiguity by defining a canonical parameter set,

$$\begin{aligned} \boldsymbol{\mu}_c &= \mathbf{K}\boldsymbol{\mu} = [1, \mu_{3x}, ..., \mu_{px}, 0, \mu_{3y}, ..., \mu_{py}]^T, \\ \boldsymbol{\Sigma}_c &= \mathbf{K}\boldsymbol{\Sigma}\mathbf{K}^T, \end{aligned} \tag{18}$$

where the mean of the second landmark has been mapped to $(1, 0)$. More study is required to see for which offset-normal shape distributions the above transformation removes all redundancies and which have a still larger set of invariant transformations. It is important to notice the difference with the non-linear mapping (3). In contrast, (18) is a linear transform, depending on $\boldsymbol{\mu}_2$. In [7] it is observed that also some singular normal pdfs or even non-normal pdfs may map to the same offset-normal shape distribution, enlarging further the redundancy. In this paper we will not concern us with those.

Transformation Properties: We will now state two important properties of the offset-normal shape distribution, which will help us derive the learning algorithm in the subsequent sections.

Lemma 1 *Let* $\mathbf{x} = [x_1, ..., x_p, y_1, ..., y_p]^T$ *be a random variable distributed according to a normal distribution with parameters* $\boldsymbol{\nu}$ *and* $\boldsymbol{\Omega}$*, and let* $\mathbf{u} = [u_3, ..., u_p, v_3, ..., v_p]^T$ *be the corresponding shape random variable, distributed according to the offset-normal shape distribution with parameters*

$$\boldsymbol{\mu} = \mathbf{L}_{p-1}\boldsymbol{\nu}, \qquad \boldsymbol{\Sigma} = \mathbf{L}_{p-1}\boldsymbol{\Omega}\mathbf{L}_{p-1}^T, \tag{19}$$

where $\mathbf{L}_{p-1}$ *is the matrix defined in (2) with the* 1*st and* $(p+1)$*st row deleted. The random variable* $\mathbf{x}' = [x'_1, ..., x'_p, y'_1, ..., y'_p]^T = \mathbf{G}\,\mathbf{x}$*, where* $\mathbf{G}$ *is a matrix of dimension* $2p \times 2p$*, will be distributed according to a* $2p$ *dimensional normal distribution with parameters* $\boldsymbol{\nu}' = \mathbf{G}\boldsymbol{\nu}$ *and* $\boldsymbol{\Omega}' = \mathbf{G}\boldsymbol{\Omega}\mathbf{G}^T$*. The corresponding shape random variables* $\mathbf{u}' = [u'_3, ..., u'_p, v'_3, ..., v'_p]^T$ *will be distributed according to an offset-normal shape distribution with parameters,*

$$\boldsymbol{\mu}' = \mathbf{L}_{p-1}\mathbf{G}\boldsymbol{\nu}, \qquad \boldsymbol{\Sigma}' = \mathbf{L}_{p-1}\mathbf{G}\boldsymbol{\Omega}\mathbf{G}^T\mathbf{L}_{p-1}^T. \tag{20}$$

The proof of this lemma is straightforward and relies on some well known properties of normal pdfs [5]. It is actually not necessary to assume that $\mathbf{G}$ is a square matrix. In general $\mathbf{G}$ can be of size $2g \times 2p$, where $2 < g \leq p$. This is useful if we want to integrate out variables from the offset-normal shape distribution [5].

Define the pair of baseline variables to be the ones which are mapped to $(0, 0)$ and $(1, 0)$. By choosing $\mathbf{G}$ to be a permutation matrix we can transform the offset-normal shape distribution between any pair of baseline variables in terms of the figure space parameters $\boldsymbol{\nu}$ and $\boldsymbol{\Omega}$. But does this still hold if we only have access to the parameters of the offset-normal shape distribution (i.e. the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$)? The following lemma answers this in the affirmative:

Lemma 2 *Let* $\mathbf{x} = [x_1, ..., x_p, y_1, ..., y_p]^T$ *be a random variable distributed according to a normal distribution with parameters* $\boldsymbol{\nu}$ *and* $\boldsymbol{\Omega}$*, and let* $\mathbf{u} = [u_3, ..., u_p, v_3, ..., v_p]^T$ *be the corresponding shape random variable, distributed according to the offset-normal shape distribution with parameters* $\boldsymbol{\mu}$ *and* $\boldsymbol{\Sigma}$*. Furthermore, let* $\mathbf{x}' = [x'_{\pi(1)}, ..., x'_{\pi(p)}, y'_{\pi(1)}, ..., y'_{\pi(p)}]^T = \mathbf{P}\mathbf{x}$ *be a permutation of* $\mathbf{x}$*, which is distributed according to a normal distribution with parameters* $\boldsymbol{\nu}' = \mathbf{P}\,\boldsymbol{\nu}$ *and* $\boldsymbol{\Omega}' = \mathbf{P}\,\boldsymbol{\Omega}\,\mathbf{P}^T$*. Then, the shape random variables* $\mathbf{u}' = [u'_{\pi(3)}, ..., u'_{\pi(p)}, v'_{\pi(3)}, ..., v'_{\pi(p)}]^T$ *are distributed according to an offset-normal shape distribution with parameters,*

$$\boldsymbol{\mu}' = \mathbf{B}\boldsymbol{\mu}, \qquad \boldsymbol{\Sigma}' = \mathbf{B}\boldsymbol{\Sigma}\mathbf{B}^T \qquad \mathbf{B} = \mathbf{L}_{p-1}\mathbf{P}\mathbf{E}, \tag{21}$$

Here $\mathbf{E}$ is the $2p \times 2p - 2$ dimensional matrix, $\mathbf{E} = \begin{bmatrix} 0 & \cdots \\ \mathbf{I} & \mathbf{0} \\ 0 & \cdots \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$. This matrix has the effect of inserting zeros at the position of the first landmark, i.e. $\boldsymbol{\mu} \rightarrow [0, \boldsymbol{\mu}_x^T, 0, \boldsymbol{\mu}_y^T]^T$ and, $\boldsymbol{\Sigma} \rightarrow \begin{bmatrix} 0 & \cdots & 0 & \cdots \\ \vdots & \boldsymbol{\Sigma}_{xx} & \vdots & \boldsymbol{\Sigma}_{xy} \\ 0 & \cdots & 0 & \cdots \\ \vdots & \boldsymbol{\Sigma}_{yx} & \vdots & \boldsymbol{\Sigma}_{yy} \end{bmatrix}$

Proof of Lemma 2 *To prove this it we need to show that the following two transformations are equivalent:*

$$\mathbf{L}_{p-1}\mathbf{P} = \mathbf{B}\mathbf{L}_{p-1} \doteq \mathbf{L}_{p-1}\mathbf{P}\mathbf{E}\mathbf{L}_{p-1} \tag{22}$$

We will multiply left and right with the identity as follows,

$$\mathbf{E}^T\mathbf{E}\mathbf{L}_{p-1}\mathbf{P} = \mathbf{E}^T\mathbf{E}\mathbf{L}_{p-1}\mathbf{P}\mathbf{E}\mathbf{L}_{p-1} \qquad \mathbf{E}^T\mathbf{E} = \mathbf{I} \tag{23}$$

Next, we notice that we can rewrite the combination $\mathbf{E}\mathbf{L}_{p-1}$ *as,*

$$\mathbf{E}\mathbf{L}_{p-1} = \mathbf{I} - \mathbf{1}\mathbf{e}_1^T \tag{24}$$

i.e. it is the $2p \times 2p$ *dimensional matrix which translates the first landmark to the origin. Using this in eqn. 23 we find,*

$$\mathbf{E}^T\left(\mathbf{P} - \mathbf{1}\mathbf{e}_1^T\mathbf{P}\right) = \mathbf{E}^T\left(\mathbf{P} - \mathbf{1}\mathbf{e}_1^T\mathbf{P}\right)\left(\mathbf{I} - \mathbf{1}\mathbf{e}_1^T\right) \tag{25}$$

Writing this out and noting that $\mathbf{P}\mathbf{1}\mathbf{e}_1^T = \mathbf{1}\mathbf{e}_1^T$ *and* $\mathbf{1}\mathbf{e}_1^T\mathbf{1}\mathbf{e}_1^T = \mathbf{1}\mathbf{e}_1^T$*, we verify that the left hand side is indeed identical to the right hand side, which then proves the lemma.* □

The relevance of this lemma is that we can compute the offset-normal shape distribution for an arbitrary pair of baseline landmarks from the offset-normal shape-parameters of a given pair of baseline landmarks. This will allow us to estimate the parameters of the shape distribution, even if the data are presented in different reference frames; a situation which may occur if one of the baseline landmarks is occluded.

In the case where we only interchange the second landmark with higher labelled landmarks, leaving the first landmark in place, the lemma slightly simplifies. In that case, $\mathbf{P}\mathbf{E} = \mathbf{E}\mathbf{P}_{p-1}$, where $\mathbf{P}_{p-1}$ is $2p-2 \times 2p-2$ dimensional permutation matrix. Therefore, using, $\mathbf{L}_{p-1}\mathbf{E} = \mathbf{I}$, we may write instead of (21),

$$\boldsymbol{\mu}' = \mathbf{P}_{p-1}\boldsymbol{\mu}, \qquad \boldsymbol{\Sigma}' = \mathbf{P}_{p-1}\boldsymbol{\Sigma}\mathbf{P}_{p-1}^T, \tag{26}$$

3 EM LEARNING ALGORITHM

Our main objective is to find parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ (or $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$) that maximize the log-likelihood of the offset-normal shape distribution given a data-set $\{\mathbf{u}_n\}$ $n = 1...N$. The log-likelihood is given by,

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{N}\sum_{n=1}^{N} \log P_{\mathcal{S}}(\mathbf{u}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{27}$$

Although the analytic form of the offset-normal shape distribution is quite complicated, the joint distribution $P(\mathbf{h}, \mathbf{u})$ is much simpler. Unfortunately, $\mathbf{h}$ is not observed and may be considered a hidden variable for that reason. This makes this estimation problem a school example of the expectation maximization (EM) algorithm. In the EM framework one iteratively optimizes the following family of objective functions (depending on the iteration k),

$$Q(k|k-1) = \frac{1}{N}\sum_{n=1}^{N} \int d\mathbf{h}\; P_{k-1}(\mathbf{h}|\mathbf{u}_n)\; \log P_k(\mathbf{h}, \mathbf{u}_n), \tag{28}$$

where $Q(k|k-1)$ depends on the parameters $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ at iteration k, *given* the parameters $\boldsymbol{\mu}_{k-1}$ and $\boldsymbol{\Sigma}_{k-1}$ at iteration $k-1$. Maximization of the log-likelihood is obtained by alternating an M-step where Q is maximized with respect to the parameters $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$, and an E-step where the posterior distribution $p(\mathbf{h}|\mathbf{u}_n)$ is determined, given the new parameters calculated in the previous M-step.

M-step: In the M-step we need to maximize $\langle \log P(\mathbf{h}, \mathbf{u}_n)\rangle_n$ with respect to $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$. Here $\langle . \rangle_n$ denotes a posterior average, $\langle f(\mathbf{h})\rangle_n = \int d\mathbf{h}\; P(\mathbf{h}|\mathbf{u}_n) f(\mathbf{h})$. The derivatives are given by,

$$\frac{\partial}{\partial \boldsymbol{\mu}_k} Q(k|k-1) = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\Sigma}_k^{-1}(\mathbf{W}_n\langle \mathbf{h}\rangle_n - \boldsymbol{\mu}_k) \tag{29}$$

$$\frac{\partial}{\partial \boldsymbol{\Sigma}_k^{-1}} Q(k|k-1) = \tag{30}$$

$$\frac{1}{2}\frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{\Sigma}_k - \mathbf{W}_n\langle \mathbf{h}\mathbf{h}^T\rangle_n \mathbf{W}_n^T + 2\mathbf{W}_n\langle \mathbf{h}\rangle_n \boldsymbol{\mu}_k^T - \boldsymbol{\mu}_k\boldsymbol{\mu}_k^T)$$

resulting in the following simple update rules:

$$\boldsymbol{\mu}_k = \frac{1}{N}\sum_{n=1}^{N} \mathbf{W}_n\langle \mathbf{h}\rangle_n \tag{31}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N}\sum_{n=1}^{N} \mathbf{W}_n\langle \mathbf{h}\mathbf{h}^T\rangle_n \mathbf{W}_n^T - \boldsymbol{\mu}_k\boldsymbol{\mu}_k^T \tag{32}$$

After every M-step we also map the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ to the canonical parameters $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$, defined in (18), to avoid drifting. Because the offset-normal shape distribution is invariant with respect to this transformation, the log-likelihood will not change either.

E-step: In the E-step we need to calculate the mean $\langle \mathbf{h}\rangle_n$ and covariance $\langle \mathbf{h}\mathbf{h}^T\rangle_n$ of the posterior distribution $P(\mathbf{h}|\mathbf{u}_n)$. Using Bayes rule it is easily found that,

$$P(\mathbf{h}|\mathbf{u}_n) = \frac{P(\mathbf{h}, \mathbf{u}_n)}{P_{\mathcal{S}}(\mathbf{u}_n)}, \tag{33}$$

where $P_{\mathcal{S}}(\mathbf{u}_n)$ is simply the offset-normal shape distribution evaluated at $\mathbf{u}_n$. Calculation of the sufficient statistics thus involves the following integrals,

$$\langle \mathbf{h}\rangle_n = \frac{1}{P_{\mathcal{S}}(\mathbf{u}_n)} \int d\mathbf{h}\; \mathbf{h}\; P(\mathbf{h}, \mathbf{u}_n) \tag{34}$$

$$\langle \mathbf{h}\mathbf{h}^T\rangle_n = \frac{1}{P_{\mathcal{S}}(\mathbf{u}_n)} \int d\mathbf{h}\; \mathbf{h}\mathbf{h}^T\; P(\mathbf{h}, \mathbf{u}_n) \tag{35}$$

These integrals can be solved following the same strategy as the one used to calculate the offset-normal shape distribution in section 2. Again, we will transform to the $\mathbf{z}$ coordinates defined in (12) and notice that,

$$\langle \mathbf{h}\rangle_n = \mathbf{R}_n\langle \mathbf{z}\rangle_n, \tag{36}$$

$$\langle \mathbf{h}\mathbf{h}^T\rangle_n = \mathbf{R}_n\langle \mathbf{z}\mathbf{z}^T\rangle_n \mathbf{R}_n^T. \tag{37}$$

Using the binomial expansion (15) and the result (17) we can calculate the following posterior averages,

$$\langle z_x^a z_y^b\rangle_n = \frac{\sum_{i=0}^{p-2} \begin{pmatrix} p-2 \\ i \end{pmatrix} \mathbf{E}_n[z_x^{2i+a}]\; \mathbf{E}_n[z_y^{2p-4-2i+b}]}{\sum_{j=0}^{p-2} \begin{pmatrix} p-2 \\ j \end{pmatrix} \mathbf{E}_n[z_x^{2j}]\; \mathbf{E}_n[z_y^{2p-4-2j}]} \tag{38}$$

Using (38) for the pairs $\{(a,b) = (1,0),(0,1),(2,0),(1,1),(0,2)\}$ allows us to perform the E-step.

Initialization: To initialize the parameters we use the approximation described in [7]. If the variances of the landmarks are small compared to the mean length of the baseline, then the offset-normal shape distribution becomes similar to a normal distribution with mean $\boldsymbol{\lambda} = [\mu_{c3x}, ..., \mu_{cpx}, \mu_{c3y}, ..., \mu_{cpy}]^T$ and covariance $\boldsymbol{\Lambda} = \mathbf{F}\boldsymbol{\Sigma}_c\mathbf{F}^T$, where $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ are are the canonical parameters and $\mathbf{F}$ is the $2p-4 \times 2p-2$ dimensional matrix,

$$\mathbf{F} = \begin{bmatrix} \lambda_1 & \mathbf{I} & \gamma_1 & \mathbf{0} \\ \vdots & & \vdots & \\ \lambda_{2p-4} & \mathbf{0} & \gamma_{2p-4} & \mathbf{I} \end{bmatrix}$$
$$\boldsymbol{\gamma} = [\mu_{3y}, ..., \mu_{py}, -\mu_{3x}, ..., -\mu_{px}]^T \quad (39)$$

To initialize our algorithm we therefore calculate the sample mean and covariance of the shape data, $\boldsymbol{\lambda} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{u}_n$ and $\boldsymbol{\Lambda} = \frac{1}{N-1}\sum_{n=1}^{N}(\mathbf{u}_n - \boldsymbol{\lambda})(\mathbf{u}_n - \boldsymbol{\lambda})^T$. The initial values of the mean $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are then given by,

$$\boldsymbol{\mu} = [1, \boldsymbol{\lambda}_x, 0, \boldsymbol{\lambda}_y] \quad (40)$$
$$\boldsymbol{\Sigma} = \mathbf{F}_+\boldsymbol{\Lambda}\mathbf{F}_+^T, \qquad \mathbf{F}_+ = \mathbf{F}^T(\mathbf{F}\mathbf{F}^T)^{-1} \quad (41)$$

where $\mathbf{F}_+$ is the pseudo-inverse of $\mathbf{F}$.

4 MIXTURE DISTRIBUTIONS

In practice it might happen that the data in figure-space are not well described by a normal distribution. In that case, we may approximate it by a mixture of Gaussians. The corresponding distribution in shape-space turns out to be a mixture of offset-normal shape distributions according to the following lemma [5]

Lemma 3 *Under a multivariate normal mixture model for the figure-space coordinates,*

$$P_{\mathbf{MoG}}(\mathbf{x}) = \sum_{a=1}^{M}\mathcal{N}_{2p}[\mathbf{x}|\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a]\,\pi_a, \quad (42)$$

the joint probability distribution function of the shape vector $\mathbf{u}$ *is a mixture of offset-normal shape distributions,*

$$P_{\mathbf{MoS}}(\mathbf{u}) = \sum_{a=1}^{M}P_{\mathcal{S}}[\mathbf{u}|\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a]\,\pi_a \quad (43)$$

The proof is simple if one realizes that every mixture component is mapped to an offset-normal shape distribution, which are then combined using the a priori probabilities π_a. To find update rules for the parameters π_a, $\boldsymbol{\mu}_a$ and $\boldsymbol{\Sigma}_a$ we start with the log-likelihood,

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}\log\sum_{a=1}^{M}P_{\mathcal{S}}(\mathbf{u}_n|a;\boldsymbol{\mu}^a, \boldsymbol{\Sigma}^a)\,\pi_a. \quad (44)$$

We will consider the labels a and the variables $\mathbf{h}$ *hidden*. The function to be iteratively maximized is therefore given by,

$$Q(k|k-1) = \quad (45)$$
$$\frac{1}{N}\sum_{n=1}^{N}\sum_{a=1}^{M}\int d\mathbf{h}\; P_{k-1}(a,\mathbf{h}|\mathbf{u}_n)\;\log\{P_k(\mathbf{h},\mathbf{u}_n|a)\;\pi_k^a\} =$$
$$\frac{1}{N}\sum_{n=1}^{N}\sum_{a=1}^{M}P_{k-1}(a|\mathbf{u}_n)\;\int d\mathbf{h}\; P_{k-1}(\mathbf{h}|\mathbf{u}_n,a)\times$$
$$\times\;\{\log P_k(\mathbf{u}_n,\mathbf{h}|a) + \log\pi_k^a\},$$

where we used, $P(\mathbf{h},a|\mathbf{u}) = P(a|\mathbf{u})\;P(\mathbf{h}|\mathbf{u},a)$. The M-step involves again maximizing this expression at every iteration with respect to π_k^a, $\boldsymbol{\mu}_k^a$ and $\boldsymbol{\Sigma}_k^a$. Taking derivatives with respect to these variables and equating them to zero we find,

$$\pi_k^a = \frac{1}{N}\sum_{n=1}^{N}P_{k-1}(a|\mathbf{u}_n), \quad (46)$$
$$\boldsymbol{\mu}_k^a = \frac{\sum_{n=1}^{N}P_{k-1}(a|\mathbf{u}_n)\;\mathbf{W}_n\langle\mathbf{h}\rangle_n^a}{\sum_{m=1}^{N}P_{k-1}(a|\mathbf{u}_m)}, \quad (47)$$
$$\boldsymbol{\Sigma}_k^a = \frac{\sum_{n=1}^{N}P_{k-1}(a|\mathbf{u}_n)\;\mathbf{W}_n\langle\mathbf{h}\mathbf{h}^T\rangle_n^a\mathbf{W}_n^T}{\sum_{m=1}^{N}P_{k-1}(a|\mathbf{u}_m)} - \boldsymbol{\mu}_k^a\boldsymbol{\mu}_k^{aT}, \quad (48)$$

where we have defined,

$$\langle f(\mathbf{h})\rangle_n^a = \int d\mathbf{h}\;P(\mathbf{h}|\mathbf{u}_n,a)f(\mathbf{h}). \quad (49)$$

These update rules are very similar to (31) and (32). In the mixture case however, the influence of every data point on $\boldsymbol{\mu}_k^a$ and $\boldsymbol{\Sigma}_k^a$ is weighted by a factor $\frac{P_{k-1}(a|\mathbf{u}_n)}{\sum_{m=1}^{N}P_{k-1}(a|\mathbf{u}_m)}$ which expresses the probability that mixture component $P(\mathbf{u}_n|a)$ is responsible for the generation of datum $\mathbf{u}_n$.

The E-step involves the calculation of $P(a|\mathbf{u}_n)$, $\langle\mathbf{h}\rangle_n^a$ and $\langle\mathbf{h}\mathbf{h}^T\rangle_n^a$. $P(a|\mathbf{u}_n)$ is simply given by,

$$P(a|\mathbf{u}_n) = \frac{P_{\mathcal{S}}(\mathbf{u}_n|a)\;\pi_a}{\sum_{b=1}^{M}P_{\mathcal{S}}(\mathbf{u}_n|b)\;\pi_b}, \quad (50)$$

where $P_{\mathcal{S}}(\mathbf{u}_n|a)$ is an offset-normal shape distribution with parameters $\boldsymbol{\mu}^a$ and $\boldsymbol{\Sigma}^a$. According to (49), the calculation of $\langle\mathbf{h}\rangle_n^a$ and $\langle\mathbf{h}\mathbf{h}^T\rangle_n^a$ is identical to those described in section 3, where we use parameters $\boldsymbol{\mu}^a$ and $\boldsymbol{\Sigma}^a$ for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. We thus see that the learning rules for a mixture of offset-normal shape distributions are straightforward generalizations of the one component learning rules.

5 INCOMPLETE DATA

In practice, it may happen that landmarks are occluded and only incomplete data are provided. First, we will assume

that the missing information does not concern the baseline points (i.e. landmarks 1 and 2). This will be generalized to arbitrary missing landmarks later in this section.

Assume that we have N, possibly incomplete samples, $\{\mathbf{u}_n\},\ n = 1...N$. For every sample we define an index m denoting the missing dimensions, and an index o denoting the observed dimensions. We will always assume that both the x and the y component of a landmark are missing, implying that m and o are necessarily even dimensional. We thus have $\mathbf{u}_n = [\mathbf{u}_n^m, \mathbf{u}_n^o]^T$ (the dependence of m and o on n is omitted for notational convenience). The question we want to answer is; *Can we use the information of incomplete data-vectors in the estimation of the parameters of the offset-normal shape distribution ?* To answer this, we first write the log-likelihood,

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{N} \sum_{n=1}^{N} \log P_{\mathcal{S}}(\mathbf{u}_n^o | \boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{51}$$

which now only depends on the observed data. This implies that we may treat the missing dimensions as hidden variables, alongside the variable $\mathbf{x}_2^*$. Thus, for every n, we have a different set of hidden variables, denoted by $\mathbf{h}_n = [\mathbf{x}_{2,n}^*, \mathbf{u}_n^m]$. In fact, it turns out to be more convenient to represent the unobserved landmarks in figure space, so that the set of missing variables becomes $\mathbf{h}_n = [\mathbf{x}_{2,n}^*, \mathbf{x}_n^{*m}]$. The auxiliary functions $Q(k|k-1)$ in terms of the above variables are given by,

$$Q(k|k-1) = \frac{1}{N} \sum_{n=1}^{N} \int d\mathbf{h}\ P_{k-1}(\mathbf{h}|\mathbf{u}_n^o)\ \log P_k(\mathbf{h}, \mathbf{u}_n^o). \tag{52}$$

The formula for $P(\mathbf{h}, \mathbf{u}^o)$ is very similar to (4) with 2 important differences. Firstly, since more variables are defined in figure space, the Jacobian of the transformation is slightly different,

$$|\det \mathbf{J}| = (x_2^{*2} + y_2^{*2})^{p-2-q}, \tag{53}$$

where q denotes the number of missing landmarks (which may be different for each data case n). Assuming for a moment that the missing dimensions have the lowest indices (i.e. $m = 3, 4, \ldots$), we define,

$$\mathbf{W}^T = \begin{bmatrix} 1 & 0 & 0 & \cdots & u_{q+1} & \cdots & u_p & 0 & 0 & 0 & \cdots \\ 0 & 1 & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots \\ \vdots & & & & & & & \vdots & & & \\ 0 & 0 & 0 & \cdots & -v_{q+1} & \cdots & -v_p & 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots \\ \vdots & & & & & & & \vdots & & & \end{bmatrix} \begin{bmatrix} \cdots & v_{q+1} & \cdots & v_p \\ \cdots & 0 & \cdots & 0 \\ & & & \vdots \\ \cdots & u_{q+1} & \cdots & u_p \\ \cdots & 0 & \cdots & 0 \\ & & & \vdots \end{bmatrix} \tag{54}$$

To generalize this to arbitrary missing dimensions we simply need to permute the columns of $\mathbf{W}^T$.

The M-step of the EM algorithm proceeds exactly as explained in section (3), where averages are now taken w.r.t. the posterior distribution $P(\mathbf{h}|\mathbf{u}_n^o)$. Evaluating these averages, which is part of the E-step, proceeds analogously as in section (3). Using equations (34) and (35) we note that the difficult part of that calculation is computing the following expectations,

$$\int d\mathbf{h}\, f(\mathbf{h}) P(\mathbf{h}, \mathbf{u}^o) = C\, \mathbf{E}[f(\mathbf{h})\, (h_1^2 + h_{q+1}^2)^{2-p+q} |\, \boldsymbol{\xi}, \boldsymbol{\Gamma}], \tag{55}$$

where $\mathbf{E}[.\ |\ \boldsymbol{\xi}, \boldsymbol{\Gamma}]$ denotes taking the average over a multivariate normal pdf with mean $\boldsymbol{\xi}$ and covariance $\boldsymbol{\Gamma}$ and $f(\mathbf{h}) = \mathbf{h}$ or $f(\mathbf{h}) = \mathbf{h}\mathbf{h}^T$. Unfortunately, the transformation in eqn. (12) will not leave the Jacobian invariant, since

$$h_1^2 + h_{q+1}^2 = \mathbf{h}^T \boldsymbol{\Omega} \mathbf{h} \qquad \boldsymbol{\Omega} = \mathbf{e}_1 \mathbf{e}_1^T + \mathbf{e}_{q+1} \mathbf{e}_{q+1}^T \tag{56}$$

is not invariant with respect to $\mathbf{h} \rightarrow \mathbf{z} = \mathbf{R}^T \mathbf{h}$. However, if we transform,

$$\begin{aligned} \boldsymbol{\Gamma} &= \mathbf{R}\mathbf{D}\mathbf{R}^T \doteq \mathbf{F}\mathbf{F}^T \\ \boldsymbol{\zeta} &= = \mathbf{F}^{-1}\boldsymbol{\xi} = \mathbf{U}\mathbf{D}^{-\frac{1}{2}}\mathbf{R}^T\boldsymbol{\xi} \\ \mathbf{z} &= \mathbf{F}^{-1}\mathbf{h} \end{aligned} \tag{57}$$

then the normal distribution transforms to, $\mathcal{N}_{\mathbf{h}}[\boldsymbol{\xi}, \boldsymbol{\Gamma}] \rightarrow \mathcal{N}_{\mathbf{z}}[\boldsymbol{\zeta}, \mathbf{I}]$ while we can still choose the orthonormal matrix $\mathbf{U}$ such that the Jacobian remains as simple as possible,

$$\mathbf{h}^T \boldsymbol{\Omega} \mathbf{h} = \mathbf{z}^T \mathbf{F}^T \boldsymbol{\Omega} \mathbf{F} \mathbf{z} = \mathbf{z}^T \boldsymbol{\Lambda} \mathbf{z} \tag{58}$$

The matrix $\boldsymbol{\Lambda}$ can be chosen diagonal by using the following eigenvalue decomposition, $\mathbf{F}^T\boldsymbol{\Omega}\mathbf{F} = \mathbf{V}\mathbf{H}\mathbf{V}^T$ which is always possible because $\mathbf{F}^T\boldsymbol{\Omega}\mathbf{F}$ is a symmetric rank-2 matrix. Thus, by choosing $\boldsymbol{\Lambda} = \mathbf{H}$ and $\mathbf{U} = \mathbf{V}^T$ we obtain the desired result. We now need to expand the Jacobian in a binomial series expansion and use eqns. (34) and (35) to arrive at an expression for the desired averages similar to eqn. (38).

Alternatively, a good approximation can be obtained by sampling from the normal distribution $\mathcal{N}[\mathbf{h}\ |\ \boldsymbol{\xi}, \boldsymbol{\Gamma}]$ and subsequent calculation of the sample average.

Missing Baseline Landmarks: Next, we treat the case where one or both of the baseline landmarks is missing from the data. For such a data case, the locations of the other landmarks should be represented in a different reference frame, i.e. using a different (observed) baseline pair. In that frame, the situation reduces to the case treated above. It remains to be understood how to incorporate data in different reference frames in the estimation process. We will first choose one, arbitrary, baseline pair and invoke lemma 2 (section 2) to write the distribution in any other

frame as,

$$P_{\mathcal{S}}(\mathbf{u}'|\,\boldsymbol{\mu}', \boldsymbol{\Sigma}') = P_{\mathcal{S}}(\mathbf{u}'|\mathbf{B}\boldsymbol{\mu}, \mathbf{B}\boldsymbol{\Sigma}\mathbf{B}^T), \qquad (59)$$

where $\mathbf{B} = \mathbf{L}_{p-1}\mathbf{P}\mathbf{E}$ and $\mathbf{L}_{p-1}$, $\mathbf{P}$ and $\mathbf{E}$ are defined in section 2. Since every data point may be defined in a different reference frame, $\mathbf{B}$ depends on n. Taking derivatives with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ in the M-step then generates the following update rules,

$$\boldsymbol{\mu}_k = \frac{1}{N}\sum_{n=1}^{N} \mathbf{B}_n^{-1}\mathbf{W}_n\langle\mathbf{h}\rangle_n \qquad (60)$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N}\sum_{n=1}^{N} \mathbf{B}_n^{-1}\mathbf{W}_n\langle\mathbf{h}\mathbf{h}^T\rangle_n\mathbf{W}_n^T\mathbf{B}_n^{-T} - \boldsymbol{\mu}_k\boldsymbol{\mu}_k^T, \qquad (61)$$

where $\mathbf{W}_n$ and $\langle . \rangle_n$ are defined in their own reference frame.

In the E-step we compute the posterior mean and covariance as usual, using parameters $\boldsymbol{\mu}'_n = \mathbf{B}_n\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}'_n = \mathbf{B}_n\boldsymbol{\Sigma}\mathbf{B}_n^T$ for data case n.

6 EXPERIMENTS

To test the algorithm on real world data, we downloaded 5 data-sets from the web [1]. Some data-sets contain data directly in shape space, while others have data in figure space, which we converted to shape space by mapping two landmarks to locations $(0, 0)$ and $(1, 0)$ respectively. Before transforming to shape space we extracted the sample mean and covariance to establish 'ground truth', since these are the parameters which describe the offset-normal shape distribution. Note however that many different normal distributions map to the same offset-normal shape distribution, so that comparing the parameters directly is not very meaningful.

Figure 1 shows the results when the sample mean and covariance were available in figure space. The data-sets used in Figure 1 are "Brizalina", 'Globorotalia' (described in [4]) and 'Mouse vertebrae' (Small group) (described in [9]). Figure 2 shows the results on the datasets 'Gorilla skulls' (female) (described in [9]) and 'Rat calvarial growth' (studied in [4]). These data-sets are defined in shape space, which implies that we have no access to ground truth. Finally, in figure 3, we present an example where we artificially generated 100 samples from a 'challenging' offset-normal shape distribution.

The algorithm usually converges within 20 iterations. Notice however, that for every data-point a SVD needs to be computed, resulting in unfavorable scaling behavior for large amounts of data.

[1] The data-sets can be found at:
http://www.amsta.leeds.ac.uk/~iand/Shape-S/datasets.html
http://life.bio.sunysb.edu/morph/index.html

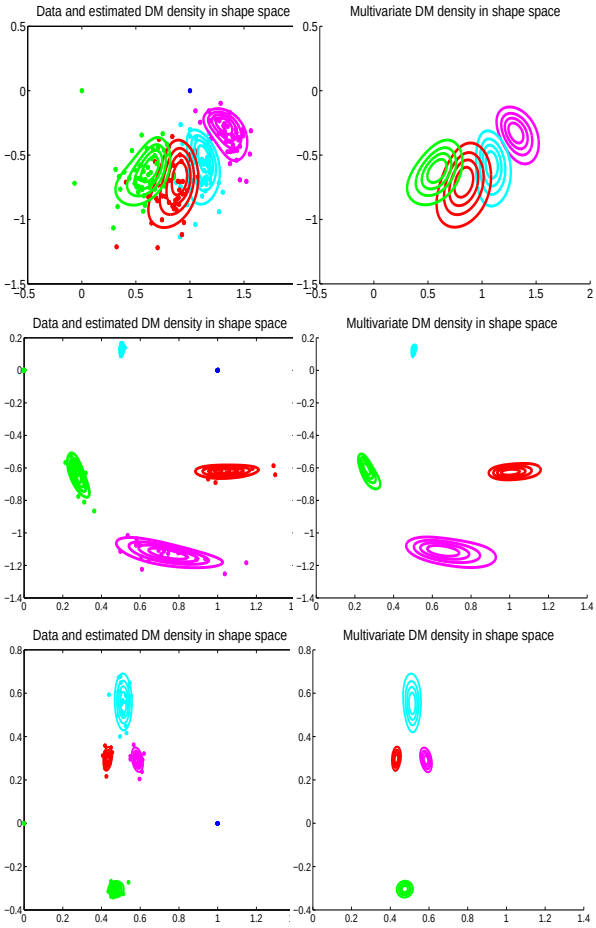


Figure 1: Estimation of offset-normal shape distributions for the following data-sets provided in figure space (from top to bottom): "Brizalina", "Globorotalia" and "Mouse vertebrae (small group)". The first column depicts the data overlaid with the offset-normal distributions estimated in shape space, while the second column shows the offset-normal distributions estimated in figure space.

We have encountered no problems in the estimation of the full covariance matrix, as described in [7], [9]. Also, few data are needed to find a reliable estimate of the distribution (around 20).

7 DISCUSSION

In this paper we have shown how to infer the parameters of a full covariance offset-normal shape distribution using the expectation maximization algorithm. In addition, we have addressed to important issues which open the door to practical applications. Firstly, the data may not be well described by an offset-normal shape distribution and secondly, the data may be incomplete, e.g. due to occlusion. The first problem was addressed by providing a learning

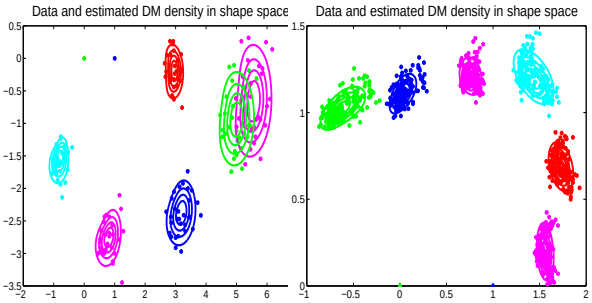


Figure 2: Estimation of offset-normal shape distributions for the following data-sets provided in shape space (left to right): "Gorilla Skulls (female)" and "Rat calvarial growth" (small group). These data-sets are only provided in shape-space, so no figure space estimates are available.

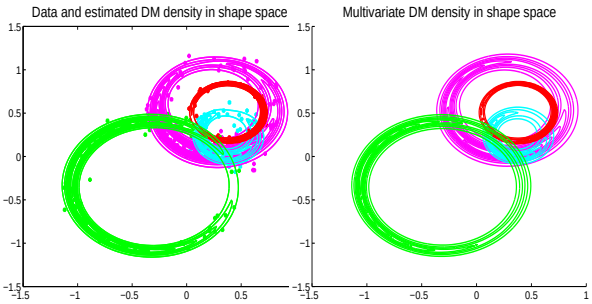


Figure 3: As in figure 1 with artificially generated data.

algorithm for mixtures of offset-normal shape distributions which improves model flexibility. The second issue was addressed by showing how to incorporate incomplete data into the estimation process.

We think the presented learning algorithms could find important applications in the field of object (class) and pattern recognition. In [6] a face recognition system was proposed where the geometry of certain feature detectors (e.g. eye-corner, nose) was described by the offset-normal shape distribution. This model also accounts for uncertainties in the labelling and the positions of the landmarks. The parameters of that model were determined in *figure* space. This was possible only because the data were acquired under carefully controlled circumstances. In more realistic situations, we want to learn the model using (possibly incomplete) shape data, which is precisely the topic of the present paper.

An important generalization of the offset-normal shape distribution is the affine invariant shape distribution proposed in [13]. There, a third landmark is mapped to a fixed position (e.g. $(x, y) = (0, 1)$), rendering the resulting distribution invariant with respect to affine transformations. The presented EM algorithm is easily extended to cover that case as well, which will be described in a future publication.

ACKNOWLEDGEMENTS

We thank CNSE and the sloan foundation for financial support. We are also grateful for discussions with Pietro Perona, Mike Burl and Markus Weber.

References

[1] F.L. Bookstein. *Lecture Notes on Biomathematics, Vol. 24*. Springer Verlag, 1978.

[2] F.L. Bookstein. A statistical method for biological shape comparison. *J. Theor. Biol.*, 107:475–520, 1984.

[3] F.L. Bookstein. Size and shape spaces for landmark data in two dimensions. *Statistical Science*, 1(2):181–242, 1986.

[4] F.L. Bookstein. *Morphometric tools for landmark data*. Cambridge University Press, 1991.

[5] M.C. Burl. *Recognition of visual object classes*. PhD thesis, Department of Electrical Engineering, California Institute of Technology, Pasadena, CA, 1997.

[6] M.C. Burl, T.K. Leung, and P. Perona. "Face Localization via Shape Statistics". In *Int Workshop on Automatic Face and Gesture Recognition*, 1995.

[7] I.L. Dryden and K.V. Mardia. General shape distributions in a plane. *Advanced Applied Probability*, 23:259–276, 1991.

[8] I.L. Dryden and K.V. Mardia. Size and shape analysis of landmark data. *Biometrika*, 79:57–68, 1992.

[9] I.L. Dryden and K.V. Mardia. *Statistical shape analysis*. Wiley, 1998.

[10] D.G. Kendall. The diffusion of shape. *Advances Applied Probability*, 9:428–430, 1977.

[11] D.G. Kendall. Shape manifolds, procrustean metrics, and complex projective spaces. *Bull. London Math Soc.*, 16:81–121, 1984.

[12] D.G. Kendall. A survey of the statistical theory of shape. *Statistical Science*, 4(2):87–120, 1989.

[13] T.K. leung, M.C. Burl, and P. Perona. Probabilistic affine invariants for recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 1998.

[14] K.V. Mardia and I.L. Dryden. Shape distributions for landmark data. *Adv. Appl. Prob.*, 21:742–755, 1989.

[15] H. Ziezold. On expected figures and a strong law of large numbers for random elements in quasi-metric spaces. In *Trans. 7th Praque Conf. Information Theory, Statistical Decision Functions, Random Processses and of the 1974 European Meeting of Statisticians*, volume A, pages 591–602, Prague, 2000.

Learning in Markov Random Fields with Contrastive Free Energies

Max Welling
School of Information and Computer Science
University of California Irvine
Irvine CA 92697-3425 USA
welling@ics.uci.edu

Charles Sutton
Department of Computer Science
University of Massachusetts
Amherst, MA 01002
casutton@cs.umass.edu

Abstract

Learning Markov random field (MRF) models is notoriously hard due to the presence of a global normalization factor. In this paper we present a new framework for learning MRF models based on the *contrastive free energy* (CF) objective function. In this scheme the parameters are updated in an attempt to match the average statistics of the data distribution and a distribution which is (partially or approximately) "relaxed" to the equilibrium distribution. We show that maximum likelihood, mean field, contrastive divergence and pseudo-likelihood objectives can be understood in this paradigm. Moreover, we propose and study a new learning algorithm: the "k-step Kikuchi/Bethe approximation". This algorithm is then tested on a conditional random field model with "skip-chain" edges to model long range interactions in text data. It is demonstrated that with no loss in accuracy, the training time is brought down on average from 19 hours (BP based learning) to 83 minutes, an order of magnitude improvement.

1 INTRODUCTION: LEARNING MRFs

In the context of machine learning two classes of graphical model have been extensively studied: the directed graphical model or Bayesian network (BN) and the undirected graphical model or Markov random field (MRF). While both models have been applied successfully in a number of domains, it is fair to say that learning in BNs has reached a more advanced level of sophistication than learning in MRFs. For instance, hidden variable models can be efficiently tackled with the variational EM algorithm[1], Bayesian inference is often feasible with conjugate priors and greedy structure learning algorithms have met with some success as well. In contrast, even for a fully observed MRF model, evaluating the gradient of the log-likelihood is typically intractable. The problem can be traced back to the presence of a *global* normalization term which depends on the parameters and which translates into an often intractable inference problem when we compute its gradient[2]. Clearly, introducing unobserved random variables only aggravates this problem, while Bayesian approaches to infer posterior distributions over parameters or structures seem completely absent in the literature, apart from one paper [9]. Because MRF models arise in many applications, including spatial statistics, computer vision, and natural-language processing, we feel that it is important to improve this state of affairs.

We claim that learning MRFs is so difficult because the inference problem induced by the global normalizer is of a different nature and often harder than the problem of computing the posterior distribution of the hidden variables *given* the observed variables needed for learning BNs. The reason is that in the latter case we enter evidence to the model and we may have reasonable hope that the posterior is peaked around a single solution. However, for MRFs we need to infer the distribution when all variables are unconstrained implying that the distribution we are trying to infer is likely to have many modes. Even though much progress has been in the field of approximate inference, no method can satisfactorily deal with a large number of modes for which the location is unknown.

To approximate the required averages over the unconstrained (model) distribution we could for instance run a MCMC sampler or use the mean field approximation [10]. While the first method is relatively slow (we need to sample for every iteration of gradient descent), the estimated statistics can also get swamped by the sampling variance[3].

[1] Fully observed BNs are trivial and only depend on counts.

[2] In case the structure of the graph is such we can identify a junction tree with small tree-width, then inference can be performed tractably and we can compute exact learning rules.

[3] Of course, one can reduce the variance by using more samples, but note that this only improves as $1/N$ where N is the number of samples.

The mean field approximation is not plagued by variance, but unlike the MCMC sampler it has to tolerate a certain bias in its estimates. However, both problems suffer from a much more severe problem, namely that they will only approximate one mode of the distribution. One could argue that a "good sampler" should mix between modes, but in the absence of any information about the location of these modes, this is an unrealistic hope, certainly in high dimensions.

There is one piece of information which typically remains unexploited, namely the fact that data points are expected to be located close to a mode (or at least this is what we like to achieve during learning). Hence, one idea to deal with the above mentioned "many modes" problem, is to run multiple MCMC chains, each one initialized at a different data-point. With this method, we are at least certain that all the modes close to data points are explored by samples. This will have the effect that learning is likely to get the local shape of each local mode correct. Still, there are (at least) two drawbacks: 1) the modes do not communicate, i.e. we have no mixing between modes and 2) accidental modes which are created because of the particular parameterization of the model remain undetected by samples implying there is no force to remove them from the model. The first problem has the undesirable effect that although the shape of each mode may be a good fit, the relative volume (or free energy) of the modes may not be properly estimated. This was studied in [7] and mode-jumping MCMC procedures were proposed to improve the communication between modes. Since there is no information about the location of the spurious modes (mentioned under 2), we predict it will be extremely hard to deal with the second problem.

Running Markov chains to convergence at every data case at every iteration of learning is clearly a costly business. Fortunately, it turns out that we can greatly improve our efficiency by running these Markov chains for only a few (say k) steps[4]. It turns out that if one uses these pseudo-samples, or rather "k-step reconstructions" of the data, we approximately minimize the so-called "contrastive divergence" objective function [5]. Apart from a very significant increase in efficiency, we also decrease the variance of our estimates at the expense of an increased bias.

The aim of the current paper is to combine deterministic, variational approximations with the ideas of contrastive divergence. This idea is analogous to the introduction of mean field learning in MRFs in [10]. A mean field based approach to contrastive divergence was presented in [17]. In the current work we extend these ideas to general variational approximations. In particular we study the Bethe approximation, which in combination with the convergent "belief optimization" algorithm to minimize the Bethe free energy results in a novel algorithm to train Markov random fields with loopy structure. This algorithm is tested on a conditional random field model with long range interactions (the so called "skip-chain" CRFs [11]) to label tokens in email messages. We demonstrate that we can speed up learning tenfold at no cost to the test-performance of the trained model.

[4] It is essential that the chains are started at the data-cases.

2 MAXIMUM LIKELIHOOD LEARNING

An intuitive way to restate the maximum likelihood objective is as a minimization problem of the following Kullback-Leibler divergence between the data distribution $P_0(y)$ and the model distribution $P_\lambda(y)$,

$$\lambda^{ML} = \arg\min_\lambda KL\left[P_0(y)||P_\lambda(y)\right] \tag{1}$$

We will consider the general case here, where apart from the observed variables, y, the model may also contain a number of unobserved variables h. Introducing the joint distribution $P_\lambda(y,h)$ and the distribution $P_0(y,h) = P_\lambda(h|y)P_0(y) = P_\lambda(h,y)P_0(y)/P_\lambda(y)$ with $P_\lambda(y) = \sum_h P_\lambda(y,h)$, we can rewrite the KL divergence as a difference between two free energies,

$$\begin{aligned} KL[P_0(y)||P_\lambda(y)] &= KL[P_0(y,h)||P_\lambda(y,h)] \\ &= F_0 - F_\infty \doteq \mathbf{CF}_\infty \geq 0 \end{aligned} \tag{2}$$

where F_0 denotes the free energy of the distribution $P_0(y,h)$, while $F_\infty = -\log(Z)$ denotes the free energy of the "random system" governed by P_λ. The subscript ∞ indicates that we have to run a Markov chain infinitely long to reach equilibrium. For every data-case we can therefore identify two random systems; one system with free energy F_0 has a data case clamped to the observed random variables while the hidden variables are free to fluctuate. In the "free system" (with free energy F_∞) all random variables (y,h) are unconstrained. The energy of the system, $E(y,h)$, is defined through the Boltzman distribution,

$$P(y,h) = \frac{1}{Z}\exp\left[-E(y,h)\right]. \tag{3}$$

Although our discussion is more general, we will restrict ourselves from now on to exponential family distributions defined through the following energy function,

$$E(y,h) = -\sum_\beta \sum_i \lambda_{i\beta} f_{i\beta}(y_\beta, h_\beta). \tag{4}$$

In analogy to physical systems, we can decompose the free energy in an average energy term and a entropy term,

$$F_0 = \mathbb{E}[E]_0 - H_0 \qquad F_\infty = \mathbb{E}[E]_\infty - H_\infty \tag{5}$$

where $\mathbb{E}[\cdot]_0$ denotes averaging with respect to the joint $P_0(y,h)$ and $\mathbb{E}[\cdot]_\infty$ denotes averaging with respect to the equilibrium distribution $P_\lambda(\mathbf{v},\mathbf{h})$.

Learning can now be understood as follows: for each data case we first compute the free energy F_0 of the system with the datum clamped to the observed units (this involves inference over the hidden units). Then we set the constraints on the observed variables free and let the system relax into a new distribution $P_\lambda(y,h)$ with lower free energy F_∞. If in this process the expected sufficient statistics $\mathbb{E}[f_{i\beta}]$ change we have an imperfect model and we change the parameters $\lambda_{i\beta}$ in such a way that the expected sufficient statistics are better preserved in the next iteration,

$$\frac{\partial \mathbf{CF}_\infty}{\partial \lambda_{i\beta}} = -\mathbb{E}[f_{i\beta}(y_\beta,h_\beta)]_{P_0} + \mathbb{E}[f_{i\beta}(y_\beta,h_\beta)]_{P_\lambda} \quad (6)$$

Note that this does not mean that the statistics for each data point must cancel with the equilibrium statistics; this property must only hold when averaged over all data cases.

3 APPROXIMATE ML-LEARNING

In the previous section, we wrote the likelihood function as a difference of two free energies, one of which was intractable to compute in general. In this section, we replace those free energies with approximate free energies, in a way conceptually similar to the mean field approximation introduced in [10]. The idea is to replace the objective in Eqn.2 with

$$\begin{aligned} &KL[Q_0(y,h)||P_\lambda(y,h)] - KL[Q_\infty(y,h)||P_\lambda(y,h)] \\ &= F_0^{\text{APP}} - F_\infty^{\text{APP}} \doteq \mathbf{CF}_\infty^{\text{APP}} \geq 0 \end{aligned} \quad (7)$$

where we define $Q_0(y,h) = Q(h|y)P_0(y)$ and where both $Q_0(h|y)$ and $Q(y,h)$ are approximate, variational distributions such as fully factorized mean field distributions or tree structured distributions. Typically they depend on a number of variational parameters that need to be computed by separately minimizing the respective KL-divergence terms in Eqn.7. The most important simplification that is achieved by minimizing $\mathbf{CF}^{\text{APP}}$ is the fact that the log-partition function term, $\log Z$, cancels between the two terms in Eqn.7.

An important constraint that must be satisfied by any contrastive free energy is that $F_0 \geq F_\infty$ or equivalently $\mathbf{CF} \geq 0$. The reason is that we like to change the unconstrained system with F_∞ so that on average it is similar to the constrained system with F_0. This would ensure that if we sample from P_λ the samples would be similar to the data-cases. Since both systems have the same energy function, but an unconstrained system has more entropy its free energy should be lower as well (see Eqn.5). Moreover, the cost function $F_0 - F_\infty$ wouldn't be lower bounded if F_∞ was allowed to become arbitrarily large.

As an example, let's choose the mean field approximation for $Q_0(h|y)$ and $Q_\infty(h,y)$ in Eqn.7 above,

$$Q_0(h|y) = \prod_i q_i(h_i|y) \qquad Q_\infty(y,h) = \prod_j r_j(z_j) \quad (8)$$

with $z = \{y,h\}$ and where both q and r are variational parameters satisfying $\sum_{h_i} q(h_i|y) = 1 \; \forall i$ and $\sum_{z_j} r(z_j) = 1 \; \forall j$. They are computed by minimizing their respective KL-divergence terms in Eqn.7. It is now easy to see that F_∞ is smaller than F_0, simply because it has more degrees of freedom to minimize over (in F_0 the variables y are constrained). It is convenient to imagine a process where we minimize F_∞ in two phases, first we clamp y to a data-case and minimize over h, then we set the y variables free and continue the minimization over (y,h) jointly[5]. Once we have found the variational parameters (q,r), we can update the parameters using the following gradient,

$$\frac{\partial \mathbf{CF}_\infty^{\text{APP}}}{\partial \lambda_{i\beta}} = -\mathbb{E}[f_{i\beta}(y_\beta,h_\beta)]_{Q_0} + \mathbb{E}[f_{i\beta}(y_\beta,h_\beta)]_{Q_\infty} \quad (9)$$

We only need to have access to (approximate) marginal distributions $p_\beta(y_\beta,h_\beta)$ in order to compute the expectations in Eqn.9. Hence, we are allowed to consider general approximate free energies F_0, F_∞ as functions of local marginal distributions only, as long as we can assert that $F_0 \geq F_\infty$. An important example of this is the family of Kikuchi free energies $F^{\text{KIK}}(\{q_\alpha\})$, where the approximate marginals need not be consistent with a global distribution Q. In other words, there may not exist a global distribution Q such that its marginals over clusters of nodes are given by the q_α which minimize F^{KIK}.

The contrastive Kikuchi free energy can be expressed as a sum over constrained local KL-divergences as follows,

$$\begin{aligned} &\mathbf{CF}_\infty^{\text{KIK}} \doteq F_0^{\text{KIK}} - F_\infty^{\text{KIK}} = \\ &\sum_\alpha c_\alpha KL[p_0(y_\alpha)q_\alpha(h_\alpha|y_\alpha)||p_\alpha(y_\alpha,h_\alpha)] - \\ &\sum_\alpha c_\alpha KL[r_\alpha(y_\alpha,h_\alpha)||p_\alpha(y_\alpha,h_\alpha)] \end{aligned} \quad (10)$$

where $p_\alpha(z_\alpha) = \frac{1}{Z_\alpha}\prod_{\beta\subset\alpha}\Psi_\beta(x_\beta)$, and where the set of clusters $\{\alpha\}$ consists of a number of overlapping large clusters which cover the graph such that any interaction Ψ_β fits in one of these clusters. By $p_0(y_\alpha)$ we mean the marginal data distribution over the variables[6] y in cluster α. Since this distribution is fixed, we only minimize over the q_α variables in the first term. The counting numbers c_α make sure that every variable and interaction is effectively counted once. Unlike the mean field approximation, the marginals are overlapping and are required to satisfy certain "marginalization constraints" on the intersections,

$$\sum_{z_\alpha \setminus z_\beta} r_\alpha(z_\alpha) = r_\beta(z_\beta) \quad (11)$$

and similarly for q. We refer to [19] for more details.

[5] In fact, the mean field equations, when run sequentially (one variable at a time), are a form of coordinate descent.

[6] Note that if we write (y_α,h_α) we mean all the variables y and h which reside in cluster α.

In the following we will be working with clusters consisting of edges and nodes only, called the "Bethe approximation", but we like to emphasize that the formalism is easily adapted to general Kikuchi approximations, or in fact region graph approximations [19]. The counting numbers in this case are given by,

$$c_{\text{edge}} = 1, \qquad c_{\text{node}} = 1 - \#\text{neighbors} \tag{12}$$

The approximate learning procedure is again similar to what we have seen before: first we compute the variational parameters (q_α, r_α) by minimizing the respective KL-divergence terms, and subsequently we update the parameters using the following gradients,

$$\frac{\partial \mathbf{CF}_\infty^{\text{BETHE}}}{\partial \lambda_{i\beta}} = -\mathbb{E}[f_{i\beta}(y_\beta, h_\beta)]_{q_\alpha p_0} + \mathbb{E}[f_{i\beta}(y_\beta, h_\beta)]_{r_\alpha} \tag{13}$$

where we need that $\beta \subseteq \alpha$.

When the free energies F_0^{BETHE} and F_∞^{BETHE} are convex in the variational parameters (q, r), we can use a class of algorithms under the name (generalized) belief propagation to minimize the Bethe free energies (or KL-divergences) in Eqn.10. However, the Bethe free energy is only convex under very special circumstances, e.g. when the graph has at most one loop. In general it is littered with local minima and for reasons explained before it does not deserve recommendation to run BP and end up in some random local minimum. Instead, we would like to initialize our minimization procedures on the data-cases. However, it is not clear how to efficiently find a set of messages that will produce a prescribed set of marginals, implying that we have little control over our initialization. Fortunately, algorithms have been developed that do not rely on messages but directly minimize the Bethe free energy as a function of the marginals [14, 20, 4]. In general, these algorithms iteratively construct a convex upper bound on the Bethe free energy and minimize those under the constraints of marginal consistency. Unfortunately, every constrained bound optimization step is a slow iterative algorithm with linear converge in general. Hence, if we use such an algorithm at every step of learning for every data-case we end up with a computationally very inefficient learning algorithm. For binary random variables with pairwise interactions the situation is considerably better, since it was shown in [18] that the constraints can be solved analytically, leaving only the node marginals as free variational parameters. Hence, a truly efficient learning procedure is currently only available for this case, but we are confident that efficient minimization algorithms for the more general case will be developed in the near future.

4 APPROXIMATE CONTRASTIVE FREE ENERGIES

We will now introduce a second approximation that is based on the ideas behind contrastive divergence and combine them with the variational approximations described in the previous section. This will have the effect of making the learning algorithm computationally much more efficient.

Recall our interpretation of learning using a contrastive free energy. First we compute the free energy F_0 at the data-case under consideration and compute the necessary sufficient statistics. Then we relax the constraints on the variables which were clamped to the value of the data-case and let the system reach equilibrium where we compute the values of the sufficient statistics again. The system is relaxed by "hitting" the data distribution P_0 with a transition kernel that has P_λ as its invariant distribution,

$$P_1(h, y) = \sum_{h', y'} \mathcal{K}(h, y | h', y') P_0(h', y') \tag{14}$$

$$P_\lambda = (\mathcal{K})^\infty P_0 \tag{15}$$

In practice we replace P_0 by the empirical distribution and achieve the relaxation by running MCMC sampling procedures initialized at the data cases.

The underlying idea of contrastive divergence is that we don't actually have to wait until the system has reached equilibrium, since there is much valuable information in the first few steps of this relaxation process (i.e. after a few steps of the MCMC samplers). If the population of samples have a systematic tendency to move away from the data, we can immediately correct this tendency by changing the parameters such that the probability becomes larger at the location of the data and the probability becomes smaller at the location of the samples obtained after a *brief* MCMC run,

$$\frac{\partial \mathbf{CF}_k^{\text{CD}}}{\partial \lambda_{i\beta}} = -\mathbb{E}[f_{i\beta}(y_\beta, h_\beta)]_{P_0} + \mathbb{E}[f_{i\beta}(y_\beta, h_\beta)]_{P_k} \tag{16}$$

Following these gradients downhill approximately minimizes the following contrastive divergence objective,

$$\begin{aligned} &KL[P_0(y,h)||P_\lambda(y,h)] - KL[P_k(y,h)||P_\lambda(y,h)] \\ &= F_0 - F_k \doteq \mathbf{CF}_k \geq 0 \end{aligned} \tag{17}$$

The derivative of this objective w.r.t. $\lambda_{i\beta}$ contains a term $\partial F_k / \partial \lambda_{i\beta}$ in addition to the terms in Eqn.16. However, it is usually small and rarely in conflict with the other terms in the gradient and as result it can be safely ignored [5].

Clearly, learning with contrastive divergence results in a vast improvement in efficiency. Moreover, because for each data-case there is a nearby sample we reduce the variance in the estimates of the sufficient statistic in Eqn.16 (compared to a MCMC sampler at equilibrium) but at the same time

we may have introduced bias in our estimates. However, it is not hard to show that for an infinite number of data-cases and a model that is flexible enough to contain the true model, it must be true that there is a fixed point at the correct parameter value, i.e. the first and second term in Eqn.20 will cancel. We refer to [5, 15, 21] for further details on contrastive divergence learning.

It is now a small step to argue for a procedure that combines the variational approximation of the previous section with the ideas of contrastive divergence. Instead of relaxing the free energy using sampling we will relax it by applying a minimization procedure over the variational distributions Q initialized at Q_0 or over the marginals $r_\alpha(z_\alpha)$, initialized at $p_0(y_\alpha)q_\alpha(h_\alpha|y_\alpha)$. Thus, we define the approximate "k-step" contrastive free energy as,

$$KL[Q_0(y,h)||P_\lambda(y,h)] - KL[Q_k(y,h)||P_\lambda(y,h)]$$
$$= F_0^{\text{APP}} - F_k^{\text{APP}} \doteq \mathbf{CF}_k^{\text{APP}} \geq 0 \tag{18}$$

where F_k^{APP} is a function of the variational distribution Q_k. Alternatively, in case of the Kikuchi approximation, we use Eqn.10 and replace the local marginals $r_\alpha(z_\alpha)$ with their k-step counterparts obtained after k steps of minimization on the Kikuchi free energy. Because of its definition the "k-step" contrastive free energy must be positive which, as discussed earlier, is an important constraint for the procedure to work. Taking derivatives w.r.t. to the parameters $\{\lambda_{i\beta}\}$ we find,

$$\frac{\partial \mathbf{CF}_k^{\text{APP}}}{\partial \lambda_{i\beta}} = \frac{\partial F_0^{\text{APP}}}{\partial \lambda_{i\beta}} - \frac{\partial F_k^{\text{APP}}}{\partial \lambda_{i\beta}} - \frac{\partial F_k^{\text{APP}}}{\partial Q_k}\frac{\partial Q_k}{\partial \lambda_{i\beta}} \tag{19}$$

where the last term appears because we didn't minimize the free energy and hence $\partial F_k/\partial Q_k \neq 0$ (unlike $\partial F_0/\partial Q_0 = 0$ and $\partial F_\infty/\partial Q_\infty = 0$). This term is difficult to compute, since we don't have explicit expressions for Q_k in terms of λ_i. Again, it is small and rarely in conflict with the other terms in the gradient so it can be safely ignored (see [17] for experimental evidence of this fact in the case of MF). Hence, ignoring the last term and simplifying the other terms we arrive at the gradient,

$$\frac{\partial \mathbf{CF}_k^{\text{APP}}}{\partial \lambda_{i\beta}} = -\mathbb{E}[f_{i\beta}(y_\beta, h_\beta)]_{Q_0} + \mathbb{E}[f_{i\beta}(y_\beta, h_\beta)]_{Q_k} \tag{20}$$

Of course, when we use the Kikuchi approximation we replace the global distributions Q_0 and Q_k in Eqn.20 by local marginals $q_\alpha p_0$ and $r_{\alpha,k}$ as in Eqn.13.

5 RELATION TO PSEUDO-LIKELIHOOD

We have seen that learning in MRFs can be interpreted as minimizing the difference between two free energies, one with the data clamped on the observed variables, the other one with all the variables unconstrained. Importantly, the latter free energy must always be lower than the first one. The various methods differed in the way we allowed the relaxation of the free energy to take place. We have introduced approximate relaxations using variational distributions and partial relaxations where we don't relax all the way to equilibrium. We will now see that the pseudo-likelihood estimator can also be interpreted in this framework (see also [6]).

In [1], the pseudo-likelihood (PL) was introduced to learn MRF models tractably. For a fully observed[7] MRF the PL is given by,

$$\text{PL} = \frac{1}{KN}\prod_{n=1}^{N}\prod_{k=1}^{K} p(\hat{y}_{k,n}|\hat{y}_{-k,n}) \tag{21}$$

where y_{-k} denotes all variables except variable y_k, K is the number of variables and N the number of data-cases. This objective is far more tractable than the ML criterion because it only depends on *one dimensional* normalization constants $Z_{k|-k}$. Moreover it was shown that asymptotically this estimator is consistent [3] (but less efficient in the statistical sense than the MLE). We can rewrite minus the log of this objective as a difference of two free energies,

$$KL[P_0||\prod_k P_{k|-k}] = \mathbb{E}[\sum_{i\beta} f_{i\beta}(y_\beta)]_{P_0} + \frac{1}{K}\sum_k \log Z_{k|-k}$$
$$= F_0^{\text{PL}} - F_\infty^{\text{PL}} = \mathbf{CF}^{\text{PL}} \geq 0 \tag{22}$$

where we identify the first term as the average energy and the second as the average one dimensional conditional partition functions. Since the data have no entropy, the first term is the free energy of the data F_0. The second term can be interpreted as a partially unconstrained free energy, where only one variable is relaxed at a time, conditioned on all the others and where the final result is averaged. Hence, like our partial relaxations, the PL-relaxation stays close to the data distribution since at all times we condition on all but one of the variables. The relaxed distribution for one data-case is given by the following mixture,

$$P_\lambda^{\text{PL}}(y_n) = \frac{1}{K}\sum_{k=1}^{K}\left[p_\lambda(y_{k,n}|\hat{y}_{-k,n})\prod_{j\setminus k}\delta(y_{j,n} - \hat{y}_{j,n})\right] \tag{23}$$

which has to be compared with P_λ (maximum likelihood), P_k (k-step contrastive divergence), Q_∞ (variational) and Q_k (k-step variational). It is now straightforward to derive the following gradients,

$$\frac{\partial \mathbf{CF}^{\text{PL}}}{\partial \lambda_{i\beta}} = -\mathbb{E}[f_{i\beta}(y_\beta)]_{P_0} + \mathbb{E}[f_{i\beta}(y_\beta)]_{P^{\text{PL}}} \tag{24}$$
$$= -\frac{1}{N}\sum_{n=1}^{N}(f_{i\beta}(\hat{y}_{\beta,n}) + \frac{1}{|\beta|}\sum_{k\subset\beta}\mathbb{E}[f_{i\beta}(y_k, \hat{y}_{\beta\setminus k})]_{p_{k|-k}})$$

[7] The following considerations are easily generalized to include hidden variables, but for simplicity we have chosen to illustrate our point using observed variables only.

where $|\beta|$ denotes the number of nodes in the cluster β.

In light of our interpretation of learning in MRFs, it is not hard to generalize the PL estimator to a generalized PL estimator where we allow the relaxation of larger, possibly overlapping clusters of nodes conditioned on the remaining nodes in the graph. We leave the study of these generalized PL estimators as future work.

As mentioned above, it has been shown that the PL estimator is asymptotically consistent, but is less efficient than the ML estimator. It would be interesting to see if the arguments in the PL-consistency proofs can be adapted to cover the estimators studied in this paper.

6 CONDITIONAL RANDOM FIELDS

A conditional random field (CRF) [8] is a MRF that is trained to maximize the *conditional* log-likelihood of labels, y, given input variables x,

$$\lambda^{ML} = \arg\min_{\lambda} KL\left[P_0(y|x)||P_\lambda(y|x)\right] \qquad (25)$$

That is, the variables that appear in the data are partitioned into input nodes x, which will be observed at test time, and output nodes y, which we will be asked to predict at test time. In practice, discriminatively-trained models often have advantages over generatively-trained models, including the ability to include many interdependent variables in x without needing to learn their distribution.

All of our previous considerations apply to the conditional case as well. However, it should be noted that for generatively-trained models the free energy F_∞ must be computed with all the variables free to fluctuate. In contrast, for discriminatively-trained models the free energy F_∞ has the data-case x_n clamped to the input nodes. Hence, the learning rule aims to match the average sufficient statistics of the random system with 1) both x and y clamped at the nodes (F_0) and 2) the random system with only x clamped at the nodes (F_∞). This has the important consequence that the relaxed distributions $P_\lambda(y|x_n)$ are different for every data-case, while the relaxed distributions for generatively-trained models $P_\lambda(y)$ are the same for all data-cases and it would in principle suffice to run a single MCMC procedure per learning iteration [8].

7 EXPERIMENTS

In this section, we evaluate the $\mathbf{CF}_k$ estimators presented in this paper on CRFs. The state of the art for training loopy CRFs in practice is penalized maximum-likelihood training with the expected sufficient statistics computed by BP [12, 13]. This has two difficulties: (a) If the model distribution has multiple modes BP may converge to different solutions depending on its initialization (or fail to converge altogether), and (b) it requires running BP to convergence at each step of gradient ascent on the log-likelihood, which will be very expensive. Therefore, if nothing else, we can still hope to achieve improved training time by using the k-step **CF** estimators introduced in this paper. For the experiments in this paper, we will use fully-observed training data, leaving partially observed data to future work.

Method	F_1 (2-clique)	F_1 (4-clique)
$\mathbf{CF}_5^{\text{BETHE}}$	**70.08**	74.94
$\mathbf{CF}_{10}^{\text{BETHE}}$	68.35	75.23
$\mathbf{CF}_{15}^{\text{BETHE}}$	61.80	76.51
$\mathbf{CF}_{500}^{\text{BETHE}}$	63.44	75.86
$\mathbf{CF}_{10}^{\text{MF}}$	57.91	55.91
ML^{MF}	60.98	65.31
ML^{BP}	68.19	**78.71**

Table 1: F_1 performance measure for various training methods on the 2-clique and 4-clique models.

Our data set is a collection of 485 e-mail messages announcing seminars at Carnegie Mellon University. The messages are annotated with the seminar's starting time, ending time, location, and speaker. This data set is due to Dayne Freitag [2], and has been used in much previous work. For reasons discussed in section 4, we consider here the binary problem of whether a word is a speaker name.

Often the speaker is listed multiple times in the same message. For example, the speaker's name might be included both near the beginning and later on, in a sentence like "If you would like to meet with Professor Smith…" It can be useful to find both such mentions, because different information can be in the surrounding context of each mention: for example, the first mention might be near an institution affiliation, while the second mentions that Smith is a professor.

To solve this problem, we wish to exploit that when the same word appears multiple times in the same message, it tends to have the same label. In a CRF, we can represent this by adding edges between output nodes (y_i, y_j) when the words x_i and x_j are identical and capitalized. Thus, the conditional distribution $p(y|x)$ has different graphical structure for different input configurations x. We use input nodes describing word identity, part-of-speech tags, capitalization, and membership in domain-specific lexicons; these are described in more detail elsewhere [11].

We compare training time and test performance of four dif-

[8] Note that we need to visit all modes with this Markov chain, so in practice it may be better to run multiple Markov chains initialized at various data-cases.

ferent contrastive free energies: ML^{MF}, which corresponds to maximum-likelihood training with mean-field free energy; ML^{BP}, which corresponds to maximum likelihood training with the Bethe free energy; and finally, $\textbf{CF}_k^{\text{MF}}$ and $\textbf{CF}_k^{\text{BETHE}}$, which correspond to k-step contrastive divergence with the mean-field and Bethe approximations, respectively. We compute the contrastive free energy as follows. For ML^{BP}, we use the TRP schedule for belief propagation [16], with messages initialized to 1. For ML^{MF}, we use damped fixed point equations with damping factor $\alpha = 0.1$ and uniform initialization. For $\textbf{CF}_k^{\text{MF}}$, however, we observed that iterating fixed-point equations for k steps might not decrease the free energy if they are improperly damped. Hence we have used separate damping factors for each data-case, $\alpha^{(i)}$, which are adapted to keep **CF** positive during learning.

To compute $\textbf{CF}_k^{\text{BETHE}}$, we use belief optimization; that is, we take k gradient steps on the Bethe free energy, eliminating the constraints by solving for the pairwise marginals and using the sigmoid parameterization described in [18]. The step-size for the gradient updates is determined by line search. For k-step contrastive divergence, it is essential that the optimization required to compute F_k^{BETHE} is initialized at the data cases. However, at the empirical distribution the derivative of the Bethe entropy is infinite. To avoid this problem we smooth the 0/1 empirical distribution by $\tilde{p}_{\text{SOFT}}(x_j) = |\tilde{p}_{0/1}(x_j) - \epsilon|$. In these experiments we use $\epsilon = 10^{-4}$.

We report performance with the F_1 measure on a per-token basis, that is:

$$F_1 = (2PR)/(P + R) \tag{26}$$

with P = # correct tokens / # tokens extracted and R = # correct tokens / # true tokens. We use ℓ_2 regularization with regularization parameter $\delta = 10$. All results are averaged over 5-fold cross validation.

First, we consider a *2-clique model* where all cliques are either linear chain edges (y_i, y_{i+1}), skip edges (y_i, y_j), and input edges (y_i, x_i)[9]. The parameters are tied over all instances of each clique type. For example, each linear chain edge (y_i, y_{i+1}) has the same weight w_{LC}. This sort of parameter tying is necessary in a conditional model because until we observe the input x, we do not know how many output nodes there will be or what connections they will have.

Table 1 compares the testing performance of the different training methods on the 2-clique model (first column). First, we note that both in **CF** and ML training, the Bethe approximation results in better accuracy than the mean-field approximation. This is as expected because the skip-chain model contains few short loops which is a graphical structure for which the Bethe approximation is more appropriate than the MF approximation. Second, with the Bethe free energy, using $\textbf{CF}_5^{\text{BETHE}}$ training results in comparable accuracy to ML training. This has great practical significance, because while the $\textbf{CF}_5^{\text{BETHE}}$ training used an average of 83 minutes to train, the ML training using belief propagation used over *19 hours*, which is an order of magnitude improvement.

Although the belief optimization algorithm has been developed for binary MRFs with pairwise interactions (a.k.a. Boltzmann machines), the CRF is free to contain arbitrary cliques with at most two output nodes, since the distribution $p(y|x)$ then still contains pairwise interactions only. To evaluate the practical advantages of such models, we also evaluate a skip chain model with higher-order cliques. In the 4-clique model, we add input nodes into the linear-chain and skip-chain cliques, so that we now have "linear-chain" cliques (y_i, y_{i+1}, x_i) and "skip" cliques (y_i, y_j, x_i, x_j) in addition to the input edges (y_i, x_i).

In Figure 1, we show the performance of $\textbf{CF}_k^{\text{BETHE}}$ model on the 4-clique model as a function of k (second column). For all values of k, the higher-order model performs better than the 2-clique model. Between the best 2-clique model and the best higher-order clique model, all 5 folds show improvement; averaging over the folds, the relative reduction in error is 20% (the F_1 rises from 70 to 76). For an unknown reason, the 2-clique model trained with $\textbf{CF}_{15}^{\text{BETHE}}$ hits a bad local maximum, but we do not see this behavior with a richer set of features. In the 4-clique model, ML training with BP does somewhat better than the best $\textbf{CF}_k^{\text{BETHE}}$ model, but there is substantial variance among the different training sets. None of the differences between ML^{BP} and $\textbf{CF}_k^{\text{BETHE}}$ for the 4-clique model are statistically significant (McNemar's test with $p > 0.1$). For the 2-clique model, on the other hand, $\textbf{CF}_5^{\text{BETHE}}$ training is significantly better than ML^{BP} ($p < 0.001$).

In summary, the experiments demonstrate two main points: that a k-step **CF** energy performs comparably to ML with vastly lower training time, and that belief optimization, which was developed for Boltzmann machines, is still effective for training models with certain higher-order cliques in a conditional setting.

8 CONCLUSION

In this paper we have offered a new view of parameter learning in MRF models as a minimization of contrastive free energies. We have seen that many objectives for MRF learning, including the likelihood function, the mean field learning objective, the contrastive divergence and the pseudo-likelihood can be written as a positive difference between two free energies. During learning we first infer the (posterior) distribution of the hidden variables given a

[9] To make the exposition simpler, we describe the models as if the only input variables x_i are the words at time i. In reality, each x_i is a vector of the observational tests described in [11].

clamped data-vector, then we relax this system (exactly, approximately or partially) by un-constraining the observed random variables. Finally we update the parameters by computing the difference of the average sufficient statistics. Not only is this unifying framework conceptually interesting, it also naturally suggests hybrid schemes where distributions are relaxed partially *and* approximately. In particular, we have studied a new learning algorithm based on the contrastive Kikuchi/Bethe free energy and its accompanying minimization algorithm, "belief optimization".

We feel that the view presented here is a rich breeding ground for new approximate learning algorithms. In future studies we hope to characterize the estimators proposed here by their asymptotic properties such as consistency and statistical efficiency.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0326249. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor. Max Welling likes to thank G. Hinton, Y.W. Teh and S. Osindero for numerous discussions on the topic.

References

[1] J. Besag. Efficiency of pseudo-likelihood estimation for simple Gaussian fields. *Biometrika*, 64:616–618, 1977.

[2] Dayne Freitag. *Machine Learning for Information Extraction in Informal Domains*. PhD thesis, Carnegie Mellon University, 1998.

[3] B. Gidas. Consistency of maximum likelihood and pseudo-likelihood estimators for Gibbs distributions. In W. Fleming and eds. P.L. Lions, editors, *Stochastic Differential Systems, Stochastic Control Theory and Applications*. New York: Springer, 1988.

[4] T. Heskes. Stable fixed points of loopy belief propagation are minima of the Bethe free energy. In *Advances in Neural Information Processing Systems*, volume 15, Vancouver, CA, 2003.

[5] G.E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.

[6] G.E. Hinton, K. Osindero, M. Welling, and Y.W. Teh. Unsupervised discovery of non-linear structure using contrastive backpropagation, 2004. in prepration.

[7] G.E. Hinton, M. Welling, and A. Mnih. Wormholes improve contrastive divergence. In *Advances in Neural Information Processing Systems*, volume 16, 2004.

[8] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

[9] I. Murray and Z. Ghahramani. Bayesian learning in undirected graphical models: approximate MCMC algorithms. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, San Francisco, CA, 2004. Morgan Kaufmann Publishers.

[10] C. Peterson and J. Anderson. A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019, 1987.

[11] Charles Sutton and Andrew McCallum. Collective segmentation and labeling of distant entities in information extraction. Technical Report TR # 04-49, University of Massachusetts, July 2004. Presented at ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields.

[12] Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML-2004)*, 2004.

[13] Ben Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In *Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI02)*, 2002.

[14] Y.W. Teh and M. Welling. The unified propagation and scaling algorithm. In *Advances in Neural Information Processing Systems*, 2002.

[15] Y.W. Teh, M. Welling, S. Osindero, and G.E. Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research - Special Issue on ICA*, 4:1235–1260, 2003.

[16] M.J. Wainwright, T. Jaakkola, and A.S. Willsky. Tree-based reparameterization for approximate estimation on loopy graphs. In *Advances Neural Information Processing Systems*, volume 14, vancouver, Canada, 2001.

[17] M. Welling and G.E. Hinton. A new learning algorithm for mean field Boltzmann machines. In *Proceedings of the International Conference on Artificial Neural Networks*, Madrid, Spain, 2001.

[18] M. Welling and Y.W. Teh. Belief optimization for binary networks: a stable alternative to loopy belief propagation. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 554–561, Seattle, USA, 2001.

[19] J.S. Yedidia, W. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical report, MERL, 2002. Technical Report TR-2002-35.

[20] A.L. Yuille. CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation. *Neural Computation*, 14(7):1691–1722, 2002.

[21] A.L. Yuille. A comment on contrastive divergence. Technical report, Department Statistics and Psychology UCLA, 2004. Technical Report.

Robust Higher Order Statistics

Max Welling
School of Information and Computer Science
University of California Irvine
Irvine CA 92697-3425 USA
welling@ics.uci.edu

Abstract

Sample estimates of moments and cumulants are known to be unstable in the presence of outliers. This problem is especially severe for higher order statistics, like kurtosis, which are used in algorithms for independent components analysis and projection pursuit. In this paper we propose robust generalizations of moments and cumulants that are more insensitive to outliers but at the same time retain many of their desirable properties. We show how they can be combined into series expansions to provide estimates of probability density functions. This in turn is directly relevant for the design of new robust algorithms for ICA. We study the improved statistical properties such as B-robustness, bias and variance while in experiments we demonstrate their improved behavior.

1 INTRODUCTION

Moments and cumulants are widely used in scientific disciplines that deal with data, random variables or stochastic processes. They are well known tools that can be used to quantify certain statistical properties of the probability distribution like location (first moment) and scale (second moment). Their definition is given by,

$$\mu_n = \mathbb{E}[x^n] \tag{1}$$

where $\mathbb{E}[\cdot]$ denotes the average over the probability distribution $p(x)$. In practise we have a set of samples from the probability distribution and compute sample estimates of these moments. However, for higher order moments these estimates become increasingly dominated by outliers, by which we will mean the samples which are far away from the mean. Especially for heavy tailed distributions this implies that these estimates have high variance and are generally unsuitable to measure properties of the distribution.

An undesirable property of moments is the fact that lower order moments can have a dominating influence on the value of higher order moments. For instance, when the mean is large it will have a dominating effect on the second order moment,

$$\mathbb{E}[x^2] = \mathbb{E}[x]^2 + \mathbb{E}[x - \mathbb{E}[x]]^2 \tag{2}$$

The second term which measures the variation around the mean, i.e. the variance, is a much more suitable statistic for scale than the second order moment. This process of subtracting lower order information can be continued to higher order statistics. The resulting estimators are called centralized moments or cumulants. Well known higher order cumulants are skewness (third order) measuring asymmetry and kurtosis (fourth order) measuring "peakiness" of the probability distribution . Explicit relations between cumulants and moments are given in appendix A (set $\mu_0 = 1$ for the classical case). Since cumulants are functions of moments up to the same order, they also suffer from high sensitivity to outliers.

Many statistical methods and techniques use moments and cumulants because of their convenient properties. For instance they follow easy transformation rules under affine transformations. Examples in the machine learning literature are certain algorithms for independent components analysis [3, 2, 1]. A well known downside of these algorithm is their sensitivity to outliers in the data. Thus, there is a need to define robust cumulants which are relatively insensitive to outliers but retain most of the convenient properties that moments and cumulants enjoy. This will be the topic of this paper.

2 MOMENTS AND CUMULANTS

A formal definition of the relation between moments and cumulants to all orders can be given in terms the characteristic function (or moment generating function) of a probability distribution,

$$\Psi(t) = \mathbb{E}[e^{ixt}] = \sum_{n=0}^{\infty} \frac{1}{n!} \mu_n (it)^n \tag{3}$$

where the last expression follows by Taylor expanding the exponential. The cumulants can now be defined by

$$\sum_{n=0}^{\infty} \frac{1}{n!} \kappa_n (it)^n = \ln \Psi(t) \tag{4}$$

where we expand the right hand side in powers of (it) and match terms at all orders.

The generalization of the above to the multivariate case is straightforward. Moments are defined as expectations of monomials,

$$\mu_{i_1,\ldots,i_m} = \mathbb{E}[x_{i_1} \ldots . x_{i_m}] \tag{5}$$

and the cumulants are again defined through the characteristic function (see Eq.7), where in addition to the univariate cumulants we now also have cross-cumulants.

From the definition of the cumulants in terms of the moments we can derive a number of interesting properties, which we will state below. It will be our objective to conserve most of these properties when we define the robust cumulants.

Lemma 1 *The following properties are true for cumulants:*

- **I.** *For a Gaussian density, all cumulants higher than second order vanish.*
- **II.** *For independent random variables, all cross-cumulants vanish.*
- **III.** *All cumulants transform multi-linearly with respect to affine transformations.*
- **IV.** *All cumulants higher than first order are invariant with respect to translations.*

The proofs of these statements can for instance be found in [9] and are very similar to the proofs for the robust cumulants which we will present in the next section.

3 ROBUST MOMENTS AND CUMULANTS

In this section we will define robust moments and cumulants by introducing an isotropic decay factor which down-weights outliers. With this decay factor we will have introduced a preferred location and scale. We therefore make the following important assumption: *The probability density function has zero-mean and unit-variance (or covariance equal to the identity in the multivariate case)*. This can always be achieved by a linear transformation of the random variables. Analogously, data will need to be centered and sphered. One may worry that these preprocessing steps are non-robust operations. Fortunately, we can rely on an extensive body of literature [5][6] to compute robust estimates of location and scale.

As will become apparent in the following, a convenient choice for the robust moments is given by the following expression,

Definition 1 *The robust moments are given by:*

$$\mu^{(\alpha)}_{i_1 \ldots i_n} = \mathbb{E}\left[(\alpha x_{i_1}) \ldots (\alpha x_{i_n}) \frac{\phi(\alpha \mathbf{x})}{\phi(\mathbf{x})} \right] \qquad \alpha \geq 1 \tag{6}$$

where $\phi(\mathbf{x})$ is the multivariate standard normal density.

The decaying factor is thus given by $\frac{\phi(\alpha \mathbf{x})}{\phi(\mathbf{x})} = \alpha^d \exp(-\frac{1}{2}(\alpha^2 - 1)\mathbf{x}^T\mathbf{x})$, where d is the dimension of the space. In the limit $\alpha \to 1$ we obtain the usual definition of moments.

In order to preserve most of the desirable properties that cumulants obey, we will use the same definition to relate moments to cumulants as in the classical case,

Definition 2 *The robust cumulants are defined by:*

$$\sum_{n=0}^{\infty} \sum_{i_1=1}^{M} \cdots \sum_{i_n=1}^{M} \frac{1}{n!} \kappa^{(\alpha)}_{i_1 \ldots i_n} (it_{i_1}) \ldots (it_{i_n}) =$$
$$\ln\left(\sum_{m=0}^{\infty} \sum_{j_1=1}^{M} \cdots \sum_{j_m=1}^{M} \frac{1}{m!} \mu^{(\alpha)}_{j_1 \ldots j_m} (it_{j_1}) \ldots (it_{j_m}) \right) \tag{7}$$

The right hand side can again be defined as the logarithm of the moment generating function for robust moments,

$$\Psi^{(\alpha)}(\mathbf{t}) = \mathbb{E}\left[\exp(i\alpha \mathbf{x}^T \mathbf{t}) \frac{\phi(\alpha \mathbf{x})}{\phi(\mathbf{x})} \right] \tag{8}$$

The explicit relation between robust moments and cumulants up to fourth order is given in appendix A.

With the above definitions we can now state some important properties for the robust cumulants. Since we assume zero-mean and unit-variance we cannot expect the cumulants to be invariant with respect to translation and scalings. However, we will prove that the following properties are still valid,

Theorem 1 *The following properties are true for robust cumulants:*

- **I.** *For a standard Gaussian density, all robust cumulants higher than second order vanish.*
- **II.** *For independent random variables, robust cross-cumulants vanish.*
- **III.** *All robust cumulants transform multi-linearly with respect to rotations.*

Proof: I: For a standard Gaussian we can compute the moment generating function analytically giving $\Psi^{(\alpha)}(\mathbf{t}) = -\frac{1}{2}\mathbf{t}^T\mathbf{t}$, implying that $\kappa^{(\alpha)}_{i_1 i_2} = \delta_{i_1 i_2}$ and all other cumulants vanish.
II: We note that if the variables $\{x_i\}$ are independent, $\Psi^{(\alpha)}(\mathbf{t})$ factorizes into a product of expectations which the logarithm turns into a sum, each term only depending on one t_i. Since cross cumulants on the left hand side of Eq.7 are precisely those terms which contain distinct t_i, they must be zero.
III: From Eq.6 we see that since the decay factor is isotropic, robust moments still transform multi-linearly with respect to rotations. If we rotate both the moments and $\mathbf{t}$ in the right-hand side of Eq.7, it remains invariant. To ensure that the left-hand side of Eq.7 remains invariant we infer that the robust cumulants must also transform multi-linearly with respect to rotations,

$$\kappa^{(\alpha)}_{i_1 \ldots i_n} \rightarrow O_{i_1 j_1} \ldots O_{i_n j_n} \kappa^{(\alpha)}_{j_1 \ldots j_n}, \qquad \mathbf{O}\mathbf{O}^T = \mathbf{O}^T\mathbf{O} = \mathbf{I} \tag{9}$$

This concludes the proof. ⊠

4 ROBUST GRAM-CHARLIER AND EDGEWORTH EXPANSIONS

Assuming we have computed robust cumulants (or equivalently robust moments) up to a given order, can we combine them to provide us with an estimate of the probability density function? For the classical case it is long known that the Gram-Charlier and Edgeworth expansions are two possibilities [8]. In this section we will show that these expansions can be generalized to the robust case as well. To keep things simple, we will discuss the univariate case here. Multivariate generalizations are relatively straightforward.

Both robust Gram-Charlier and Edgeworth expansions will be defined as series expansions in the scaled Hermite polynomials $H_n(\alpha x)$.

$$p(x) = \sum_{n=0}^{\infty} c_n^{(\alpha)} H_n(\alpha x)\phi(x) \quad \text{with} \tag{10}$$

$$c_n^{(\alpha)} = \frac{1}{n!}\int_{-\infty}^{\infty} p(x) H_n(\alpha x)\phi^{-1}(x)\, d\nu_\alpha \tag{11}$$

where we have defined the measure $d\nu_\alpha = \phi(\alpha x)\, dx$ and used the following generalized orthogonality relation,

$$\int_{-\infty}^{\infty} H_n(\alpha x) H_m(\alpha x)\, d\nu_\alpha = n!\, \delta_{nm} \tag{12}$$

When $c_n^{(\alpha)}$ is estimated by averaging over samples (Eq.25), we see that the decay factor $\frac{\phi(\alpha x)}{\phi(x)}$ will again render them robust against outliers.

We may also express the above series expansion directly in terms of the robust cumulants. The explicit expression is given by the following theorem[1],

Theorem 2 *The series expansion of a density $p(x)$ in terms of its robust cumulants is given by*

$$p(x) = \frac{\phi(x)}{\phi(\alpha x)} e^{(\sum_{n=0}^{\infty} \frac{1}{n!}\tilde{\kappa}_n^{(\alpha)}(-1)^n \frac{d^n}{d(\alpha x)^n})}\phi(\alpha x) \tag{13}$$

$$\textit{with} \qquad \tilde{\kappa}_n^{(\alpha)} = \kappa_n^{(\alpha)} - \delta_{n,2} \tag{14}$$

Proof: see appendix B.

To find an explicit expression up to a certain order in the robust cumulants, one expands the exponential and uses $(-1)^n \frac{d^n}{dx^n}\phi(x) = H_n(x)\phi(x)$ to convert derivatives into Hermite polynomials.

Analogous to the classical literature we will talk about a Gram-Charlier expansion when we expand in $c_n^{(\alpha)}$ and an Edgeworth expansion when we expand in $\kappa_n^{(\alpha)}$. Their only difference is therefore in their convention to break the series off after a finite number of terms.

When $\alpha = 1$ the Hermite expansions discussed in this section will be normalized, even when only a finite number of terms is taken into account. This holds since $H_0 = 1$ and $c_0 = 1/N\sum_n 1 = 1$, while all higher order polynomials are orthogonal to "1". When generalizing to robust cumulants this however no longer holds true. To correct this we will add an extra term to the expansion,

$$p_R(x) = \{\sum_{n=0}^{R} c_n^{(\alpha)} H_n(\alpha x) + \psi(x)\}\phi(x), \tag{15}$$

The correction factor can be computed by a Gram-Schmidt procedure resulting in,

$$\psi(x) = (1 - \sum_{n=0}^{R} n! a_n c_n^{(\alpha)})(\frac{\phi(x)}{\phi(\alpha x)} - \sum_{n=0}^{R} a_n H_n(\alpha x)). \tag{16}$$

with $a_n = \frac{(n-1)!!}{n!}(\alpha^2 - 1)^{\frac{n}{2}}\, \delta_{n,2k}$ for $k \in \{0, 1, 2, 3, ...\}$ and $(n-1)!!$ denotes the double factorial of $(n-1)$ defined by $1 \cdot 3 \cdot 5 ... (n-1)$. The correction factor is thus orthogonal to all Hermite polynomials $H_n(\alpha x)$ with $n = 1..R$ under the new measure $d\nu_\alpha$. We can also show that $p_R(x)$ always integrates to 1 and that when $\alpha \rightarrow 1$ the correction term will reduce to $\psi(x) \rightarrow c_{R+K} H_{R+K}(x)$ with $K = 1$ when R is odd and $K = 2$ when R is even. Finally we note that since $\int_{-\infty}^{\infty} \phi^2(x)/\phi(\alpha x)dx = 1/(\alpha\sqrt{2-\alpha^2})$ the

[1]The equivalent result in the multivariate case is,

$$p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{\phi(\alpha\mathbf{x})} \times$$
$$e^{(\sum_{n=0}^{\infty}\sum_{i_1=1}^{M}\cdots\sum_{i_n=1}^{M}\frac{1}{n!}\tilde{\kappa}^{(\alpha)}_{i_1\ldots i_n}(-1)^n \frac{d}{d(\alpha x)_{i_1}}\cdots\frac{d}{d(\alpha x)_{i_n}})}\phi(\alpha\mathbf{x})$$
$$\text{with} \qquad \tilde{\kappa}^{(\alpha)}_{i_1 i_2} = \kappa^{(\alpha)}_{i_1 i_2} - \delta_{i_1 i_2}$$

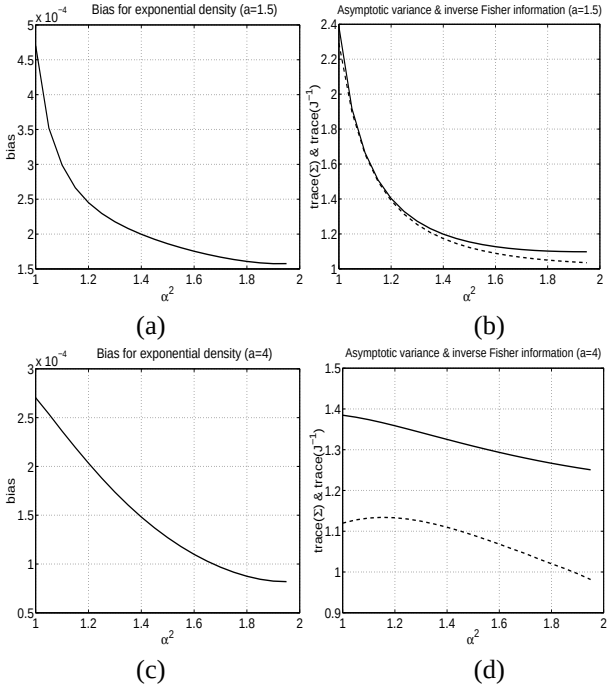


Figure 1: (a)-Bias as a function of α^2 for a generalized Laplacian with $a = 1.5$ (super-Gaussian). (b)-Asymptotic variance (solid line) and inverse Fisher information (dashed line) as a function of α^2 for $a = 1.5$. (c)-(d) Similar plots for $a = 4$ (sub-Gaussian)

correction is only normalizable for $\alpha^2 < 2$, which is what we will assume in the following.

5 CONSISTENCY, ROBUSTNESS, BIAS AND VARIANCE

In this section we will examine the robustness, bias and efficiency of our generalized expansion. Many definitions in this section are taken from [5]. Our analysis will assume that the data arrive centered and sphered, which allows us to focus on the analysis of the higher order statistics. For a thorough study of the robustness properties of first and second order statistics see [5].

First we mention that the estimators $\hat{c}_n^{(\alpha)}[p_R]$ for the truncated series expansion (Eq.15) are Fisher consistent. This can be shown by replacing $p(x)$ in Eq.11 with $p_R(x)$ and using orthogonality between $\psi(x)$ and the Hermite polynomials $H_n(\alpha x)\ \ n = 1..R$ w.r.t. the measure $d\nu_\alpha$.

To prove B-robustness we need to define and calculate the influence function IF for the estimators $\hat{c}_n^{(\alpha)}$. Intuitively, the influence function measures the sensitivity of the estimators to adding one more observation at location x,

$$IF(x) = \lim_{t\to 0} \frac{c_n^{(\alpha)}[(1-t)p_R + t\delta_x] - c_n^{(\alpha)}[p_R]}{t}. \quad (17)$$

An estimator is called B-robust if its influence function is finite everywhere. We will now state the following result.

Theorem 3 *The estimates $\hat{c}_n^{(\alpha)}[p_R]$ are B-robust for $\alpha > 1$.*

Proof: It is straightforward to compute the influence function defined in Eq.17,

$$IF(x) = \frac{1}{n!} H_n(\alpha x) \frac{\phi(\alpha x)}{\phi(x)} - c_n^{(\alpha)} \quad (18)$$

Since for $\alpha > 1$ this IF is finite everywhere, the result follows. $\boxtimes$

Since cumulants are simple functions of the $c_n^{(\alpha)}$ up to the same order, we conclude that cumulants are also B-robust. It is important to notice that in the classical case ($\alpha = 1$) the theorem does not hold, confirming that classical cumulants are not robust. Analogously one can show that the sensitivity to shifting data-points is also bounded for $\alpha > 1$.

We now turn to the analysis of bias and variance. It is well known that the point-wise mean square error can be decomposed into a bias and a variance term,

$$\mathbf{MSE}_x(p_R^{(N)}(x)) = \mathbb{E}\left[(p_R^{(N)}(x) - p(x))^2\right] = \mathbb{E}\left[(p_R^{(N)}(x) - p_R(x))^2\right] + (p_R(x) - p(x))^2 \quad (19)$$

where $p_R^{(N)}$ is the estimate of p_R using a sample of size N. The expectation $\mathbb{E}$ is taken over an infinite number of those samples. Clearly, the first term represents the variance and the second the bias which is independent of N. The variance term (V) may be rewritten in terms of the influence function,

$$V = \frac{1}{N} \sum_{n,m=0}^{R} \Sigma(c_n^{(\alpha)}, c_m^{(\alpha)}) H_n(\alpha x) H_m(\alpha x) \phi^2(x) \quad (20)$$

$$\Sigma(c_n^{(\alpha)}, c_m^{(\alpha)}) = \int_{-\infty}^{\infty} p(x) IF(x, c_n^{(\alpha)}) IF(x, c_m^{(\alpha)}) dx \quad (21)$$

So the variance decreases as $1/N$ with sample size while the data independent part is completely determined by the asymptotic covariance matrix Σ which is expressed in terms of the influence function.

Finally, by defining the Fisher information as,

$$J(c_n^{(\alpha)}, c_m^{(\alpha)}) = \mathbb{E}\left[\frac{1}{p(x)} \frac{\partial}{\partial c_n^{(\alpha)}} p_R(x) \frac{1}{p(x)} \frac{\partial}{\partial c_m^{(\alpha)}} p_R(x)\right]_p = \int_{-\infty}^{\infty} \frac{H_n(\alpha x) H_m(\alpha x) \phi(x)^2}{p(x)}\, dx \quad (22)$$

the well known Cramer-Rao bound follows: $\Sigma(c_n^{(\alpha)}, c_m^{(\alpha)}) \geq J^{-1}(c_n^{(\alpha)}, c_m^{(\alpha)})$.

In figure 1 we plot the bias and the total variation (trace of the covariance) as a function of α^2 for a super-Gaussian and a sub-Gaussian density (generalized Laplace density $p \propto \exp(-b|x|^a)$ with unit variance and $a = 1.5$ and $a = 4$ respectively) . The trace of the inverse Fisher information

was also plotted (dashed line). The model included 10 orders in the expansion $n = 0, ..., 9$ plus the normalization term $\psi(x)$. All quantities were computed using numerical integration. We conclude that *both* bias and efficiency improve when α moves away from the classical case $\alpha = 1$.

6 INDEPENDENT COMPONENTS ANALYSIS

Although robust moments and cumulants can potentially find applications in a broad range of scientific disciplines, we will illustrate their usefulness by showing how they can be employed to improve algorithms for independent components analysis (ICA). The objective in ICA is to find a new basis for which the data distribution factorizes into a product of independent one-dimensional marginal distributions. To achieve this, one first removes first and second order statistics from the data by shifting the sample mean to the origin and sphering the sample covariance to be the identity matrix. These operations render the data *decorrelated* but higher order dependencies may still remain. It can be shown [2] that if an independent basis exists, it must be a rotation away from the basis in which the data is de-correlated, i.e. $\mathbf{x}_{ica} = \mathbf{O}\mathbf{x}_{decor}$ where $\mathbf{O}$ is a rotation. One approach to find $\mathbf{O}$ is to propose a contrast function that, when maximized, returns a basis onto which the data distribution is a product of independent marginal distributions. Various contrast functions have been proposed, e.g. the neg-entropy [4] and the mutual information [1]. All contrast functions share the property that they depend on the marginal distributions which need to be estimated from the data. Naturally, the Edgeworth expansion [4, 3] and the Gram-Charlier expansion [1] have been proposed for this purpose. This turns these contrast functions into functions of moments or cumulants. However, to obtain reliable estimates one needs to include cumulants of up to fourth order. It has been observed frequently that in the presence of outliers these cumulants often become unreliable (e.g. [7]).

We propose to use the robust Edgeworth and Gram-Charlier expansions discussed in this paper instead of the classical ones. As we will show in the experiments below, it is safe to include robust cumulants to very high order in these expansions (we have gone up to order 20), which at a moderate computational cost will have a significant impact on the accuracy of our estimates of the marginal distributions. We note that the derivation of the contrast function in e.g. [4] crucially depends on properties I,II and III from theorem 1. This makes our robust cumulants the ideal candidates to replace the classical ones. Instead of going through this derivation we will argue for a novel contrast function that represents a slight generalization of the one proposed in [4],

$$I(\mathbf{O}) = \sum_{n=1}^{R} \sum_{i=1}^{M} w_n (\tilde{\kappa}^{(\alpha)}_{i...i})^2 \qquad w_n \geq 0, \qquad (23)$$

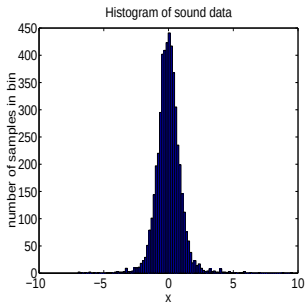


Figure 2: Histogram of sound-data (5000 samples).

where $\tilde{\kappa}^{(\alpha)}_{i...i}$ only differ from the usual $\kappa^{(\alpha)}_{i...i}$ in second order, $\tilde{\kappa}^{(\alpha)}_{ii} = \kappa^{(\alpha)}_{ii} - 1$. These cumulants are defined on the rotated axis $\mathbf{e}'_i = \mathbf{O}^T \mathbf{e}_i$.

We will now state a number of properties that show the validity of $I(\mathbf{O})$ as a contrast function for ICA,

Theorem 4 *The following properties are true for $I(\mathbf{O})$:*

- **i.** *$I(\mathbf{O})$ is maximal if the probability distribution on the corresponding axis factors into an independent product of marginal distributions.*
- **ii.** *$I(\mathbf{O})$ is minimal (i.e. 0) if the marginal distributions on the corresponding axis are Gaussian.*

Proof: To prove (i) we note that the following expression is scalar (i.e. invariant) w.r.t. rotations[2],

$$\sum_{i_1...i_n} (\tilde{\kappa}^{(\alpha)}_{i_1...i_n})^2 = constant \qquad \forall n \qquad (24)$$

We now note that this expression can be split into two terms: a sum over the "diagonal terms" where $i_1 = i_2 = \ldots = i_n$ and a sum over all the remaining cross-cumulant terms. When all directions are independent all cross-cumulants must vanish by property II of theorem 1. This minimizes the second term (since it's non-negative). Hence, by the fact the sum of these terms is constant, the first term, which equals $I(\mathbf{O})$, must be maximal for independent directions.
To prove (ii) we invoke property I of theorem 1 that for Gaussian random variables all cumulants $\tilde{\kappa}$ must vanish. $\boxtimes$

By the above theorem we see that $I(\mathbf{O})$ simultaneously searches for independent directions and non-Gaussian directions. Observe however, that for practical reasons we have ignored cumulants of order higher than R. Hence, there will certainly be more than one distribution which

[2] For vectors this reduces to the statement that an inner product is scalar. To prove the general case we use $\mathbf{O}^T\mathbf{O} = \mathbf{I}$ for every index separately.

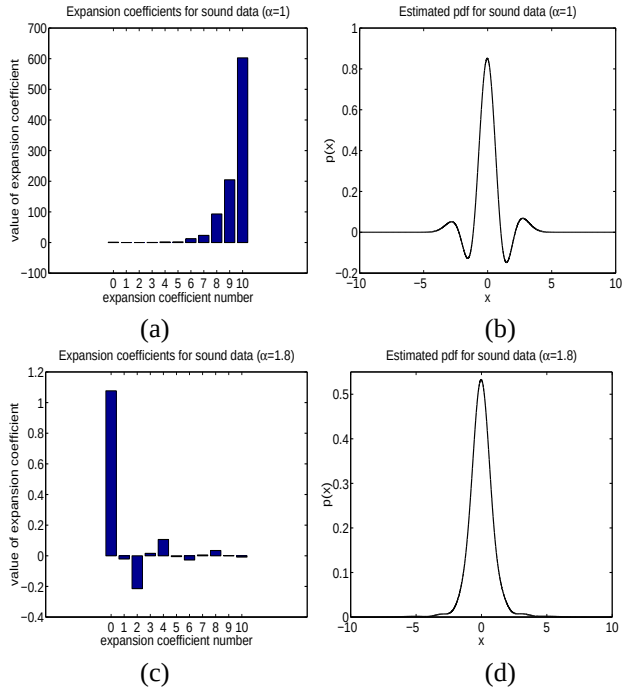


Figure 3: (a)-Expansion coefficients for classical Gram-Charlier expansion ($\alpha = 1$). (b)-Density estimate for $\alpha = 1$ after four orders. The negative tails signal the onset of a diverging series. (c)-Decreasing expansion coefficients for $\alpha = 1.8$. (f)-Density estimate after 10 orders for $\alpha = 1.8$.

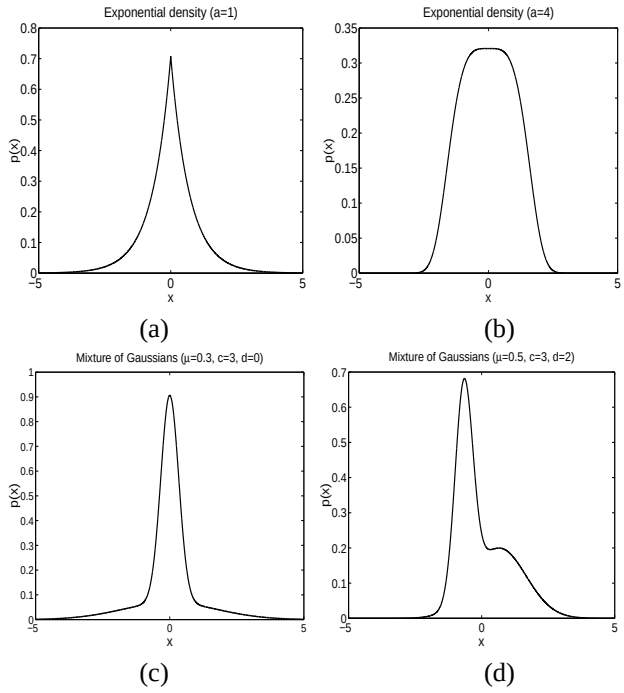


Figure 4: Top row: Generalized Laplace distributions with (a) $a = 1$, (b) $a = 4$. Bottom row: Mixture of Gaussians with (c) $\mu = 0.3$, $c = 3$, $d = 0$ and (d) $\mu = 0.5$, $c = 3$, $d = 2$.

maximizes $I(\mathbf{O})$ (for instance distributions which only differ in the statistics of order higher than R). Good objective functions are discriminative in the sense that there are only few (relevant) densities that maximize it. We can influence the ability of $I(\mathbf{O})$ to discriminate by changing the weighting factors w_n. Doing this allows for a more directed search towards predefined qualities, e.g. a search for high kurtosis directions would imply a large w_4.

A straightforward strategy to maximize $I(\mathbf{O})$ is gradient ascent while at every iteration projecting the solution back onto the manifold of rotations (e.g. see [10]). A more efficient technique which exploits the tensorial property of cumulants (i.e. property III of theorem 1) was proposed in [3]. This technique, called Jacobi-optimization, iteratively solves two dimensional sub-problems analytically.

7 EXPERIMENTS

The following set of experiments focus on density estimates based on the Gram-Charlier expansion (Eq.10) where we replace Eq.11 with a sample estimate,

$$\hat{c}_n^{(\alpha)} = \frac{1}{N}\frac{1}{n!}\sum_{A=1}^{N}\frac{\phi(\alpha x_A)}{\phi(x_A)}H_n(\alpha x_A) \qquad (25)$$

The reason we focus on this task is that we can demonstrate robustness by showing that low order robust statistics are always dominant over higher order robust statistics, even for heavy tailed distributions. Yet at the same time they carry the relevant information of the probability density, i.e. they combine into an accurate estimate of it. This exercise is also relevant for cumulant based algorithms for independent components analysis because they rely on the fact that the Gram-Charlier or Edgeworth expansions describe the source distributions well.

Sound Data

We downloaded recordings from music CD's [3] and extracted 5000 samples from it. The histogram is shown in figure 2. Due to the presence of outliers we expect the classical expansion to break down. This can be observed from figure (3a) where the coefficients *increase* with the order of the expansion. In figure (3b) we see that the density estimate has become negative in the tails after 4 orders, which is an indication that the series has become unstable. In figures (3c,d) we see that for the robust expansion at $\alpha = 1.8$ the coefficients decrease with order and the estimate of the density is very accurate after 10 orders.

Synthetic Data

In this experiment we sampled 5000 data-points from two generalized Laplace densities $p \propto \exp(-b|x|^a)$ (figures 4a,b) and from two mixtures of two Gaussians parameterized as

$p_{\mathbf{mog}}(x) = \mu a\phi(ax+b) + c(1-\mu)\phi(cx+d)$ (figures 4c,d). These include super-Gaussian distributions (figures

[3] http://sweat.cs.unm.edu/bap/demos.html

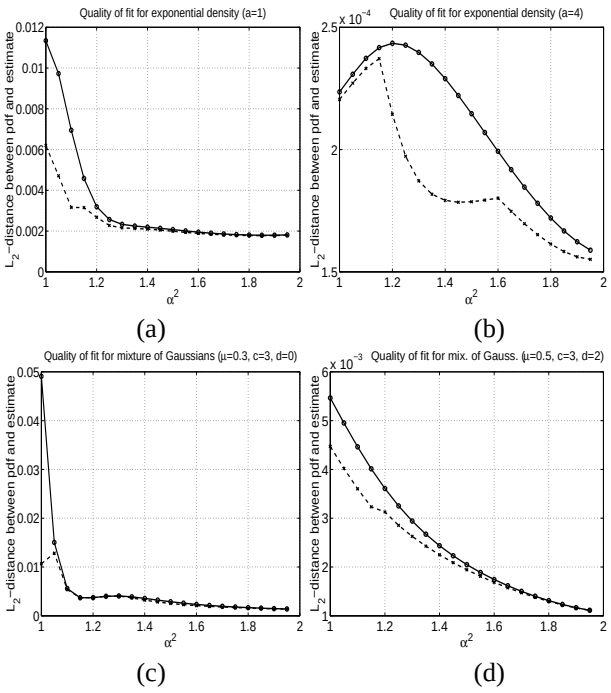


Figure 5: Top row: total L_2 distance between true and estimated densities as function of α^2 for generalized Laplace density with (a) $a = 1$, (b) $a = 4$. Bottom row: same as top row for the mixture of Gaussians distributions with (c) $\mu = 0.3$, $c = 3$, $d = 0$ and (d) $\mu = 0.5$, $c = 3$. The corresponding densities are shown in figure 4. Dashed line indicates the best estimate over all orders.

4a,c), a sub-Gaussian density (figures 4b) and an asymmetric density (figures 4d). We plot the total L_2 distance between the estimate and the true density as we vary α (figures 5a,b,c,d). Shown is the best estimate over all orders orders (dashed line) and the final estimate after 20 orders. In both cases it is observed that the best estimates are obtained around $\alpha^2 \approx 2$ (but recall that $\alpha^2 < 2$, see section 4. We also plot the L_2 distance between true and estimated density as a function of the order of the expansion for $\alpha^2 = 1$ and $\alpha^2 = 1.9$ ($a = 1$) in figures (6a,b). Clearly, the robust expansion converges while the classical expansion is unstable. Finally, in figure 7 we compare the best estimated PDFs for the general Laplace density at $a = 1$ with $\alpha^2 = 1$ (a) and $\alpha^2 = 1.9$ (b).

The general conclusion from these experiments is that in all cases (super- or sub-Gaussian PDF, symmetric or asymmetric PDF) we find that the quality (in L_2-norm) of the estimated densities improves considerably when we use the robust series expansion with a setting of α^2 close to (but smaller than) 2. This effect is more pronounced for super-Gaussian densities than for sub-Gaussian densities.

8 DISCUSSION

In this paper we have proposed robust alternatives to higher order moments and cumulants. In order to arrive at robust

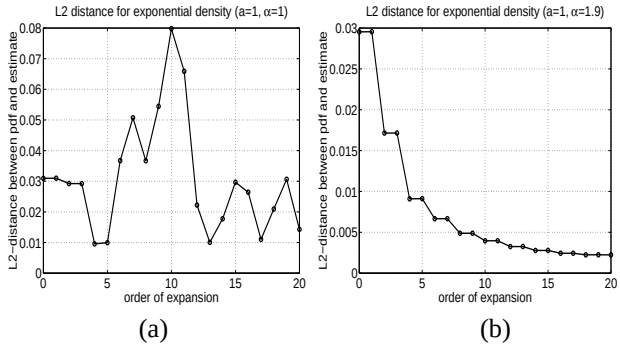


Figure 6: L_2-distance as a function of the order of the expansion for (a) $\alpha^2 = 1$ and (b) $\alpha^2 = 1.9$ for the generalized Laplace PDF with $a = 1$.

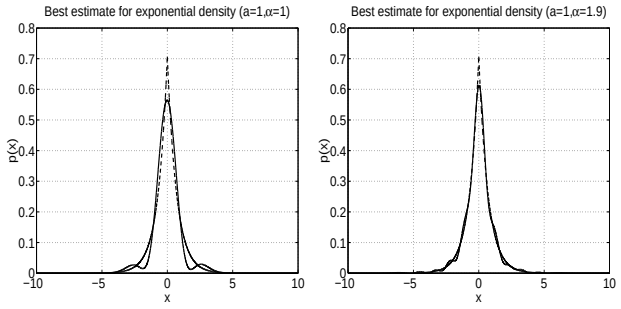
cumulants invariance w.r.t. translations was lost and the class of transformations under which they transform multi-linearly was reduced from affine to orthogonal (i.e. rotations). However, all other cumulant properties were conveniently preserved. We argue that by first centering and sphering the data (using robust techniques described in the literature [5]), multi-linearity w.r.t. orthogonal transformations is all we need, which could make the trade-off with improved robustness properties worthwhile.

There is two well-known limitations of cumulants that one needs to be aware of. Firstly, they are less useful as statistics characterizing the PDF if the mass is located far away from the mean. Secondly, the number of cumulants grows exponentially fast with the dimensionality of the problem. With these reservations in mind, many interesting problems remain, even in high dimensions, that are well described by cumulants of low dimensional marginal distributions, as the ICA example has illustrated.

The sensitivity to outliers can be tuned with the parameter $\alpha^2 \in [1, 2)$. Our experiments have shown that if one includes many orders in the expansion, optimal performance was obtained when α^2 was close to (but smaller than) 2. Although unmistakeably some information is ignored by weighting down the impact of outliers, the experiments indicated that the relevant information to estimate the PDF was mostly preserved. In future experiments we hope to show that this phenomenon is also reflected in improved performance of ICA algorithms based on robust cumulants.

A ROBUST MOMENTS AND CUMULANTS TO 4'TH ORDER

This appendix contains the definition of the cumulants in terms of the moments and vice versa for general α. We have not denoted α explicitely in the following for nota-



(a) (b)

Figure 7: Best estimates for the generalized Laplace density at $a = 1$. In (a) we plot the best classical estimate which is found after four orders of Hermite polynomials are taken into account (i.e. $H_0(x), ..., H_4(x)$). For higher orders, the series becomes unstable and the calculation of the expansion coefficients is too sensitive to sample fluctuations. The best estimate from the robust expansion is depicted in (b). In that case the best estimate is found when all orders are taken into account, i.e. 20.

tional convenience.

$$\kappa_0 = \ln \mu_0 \qquad \kappa_1 = \frac{\mu_1}{\mu_0} \qquad \kappa_2 = \frac{\mu_2}{\mu_0} - (\frac{\mu_1}{\mu_0})^2$$
$$\kappa_3 = \frac{\mu_3}{\mu_0} - 3\frac{\mu_1\mu_2}{\mu_0^2} + 2(\frac{\mu_1}{\mu_0})^3$$
$$\kappa_4 = \frac{\mu_4}{\mu_0} - 3(\frac{\mu_2}{\mu_0})^2 - 4\frac{\mu_1\mu_3}{\mu_0^2} + 12\frac{\mu_1^2\mu_2}{\mu_0^3} - 6(\frac{\mu_1}{\mu_0})^4$$
$$\mu_0 = e^{\kappa_0} \qquad \frac{\mu_1}{\mu_0} = \kappa_1 \qquad \frac{\mu_2}{\mu_0} = \kappa_2 + \kappa_1^2$$
$$\frac{\mu_3}{\mu_0} = \kappa_3 + 3\kappa_2\kappa_1 + \kappa_1^3$$
$$\frac{\mu_4}{\mu_0} = \kappa_4 + 4\kappa_3\kappa_1 + 3\kappa_2^2 + 6\kappa_2\kappa_1^2 + \kappa_1^4$$

B PROOF OF THEOREM 2

The characteristic function or moment generating function of a PDF is defined by:

$$\Psi(t) = \int_{-\infty}^{\infty} e^{ixt} p(x)\, dx = \sum_{n=0}^{\infty} \frac{1}{n!}\mu_n (it)^n = \mathcal{F}[p(x)] \tag{26}$$

where the last term follows from Taylor expanding the exponential and $\mathcal{F}$ denotes the Fourier transform. For arbitrary α we have,

$$\Psi^{(\alpha)}(t) = \int_{-\infty}^{\infty} e^{i\alpha xt} p(x) \frac{\phi(\alpha x)}{\phi(x)} dx$$
$$= \sum_{n=0}^{\infty} \frac{1}{n!}\mu^{(\alpha)}{}_n (it)^n dx = \mathcal{F}[p(x)\frac{\phi(\alpha x)}{\alpha\phi(x)}]. \tag{27}$$

Where in the last equality the definition of the generalized moments (Eq.6) was used. $\Psi^{(\alpha)}$ is the *(robust) moment generating function* of $p(x)$. We can find an expression for $p(x)$ if we invert the Fourier transform,

$$p(x) = \frac{\alpha\phi(x)}{\phi(\alpha x)} \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-i\alpha xt} \Psi^{(\alpha)}(t)\, dt. \tag{28}$$

Next, we use the relation between the cumulants and the moments (Eq.7) to write,

$$p(x) = \frac{\alpha\phi(x)}{\phi(\alpha x)} \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-i\alpha xt}\, e^{\sum_{n=0}^{\infty} \frac{1}{n!}\kappa_n^{(\alpha)}(it)^n}\, dt. \tag{29}$$

By defining $\tilde{\kappa}_n^{(\alpha)} = \kappa_n^{(\alpha)} - \delta_{n,2}$ we can separate a factor $\phi(t)$ (Gaussian) inside the integral,

$$p(x) = \frac{\alpha\phi(x)}{\phi(\alpha x)} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-i\alpha xt}\, e^{\sum_{n=0}^{\infty} \frac{1}{n!}\tilde{\kappa}_n^{(\alpha)}(it)^n} \phi(t)\, dt. \tag{30}$$

Finally, we will need the result

$$\mathcal{F}^{-1}[(it)^n \phi(t)] = \frac{\sqrt{2\pi}}{\alpha}(-1)^n \frac{d^n}{d(\alpha x)^n}\phi(\alpha x). \tag{31}$$

If we expand the exponential containing the cumulants in a Taylor series, and do the inverse Fourier transform on every term separately, after which we combine the terms again in an exponential, we find the desired result (Eq.14).

References

[1] S. Amari, A. Cichocki, and H.H. Yang. A new algorithm for blind signal separation. *Advances in Neural Information Processing Systems*, 8:757–763, 1996.

[2] A.J. Bell and T.J. Sejnowski. The independent components of natural scenes are edge filters. *Vision Research*, 37:3327–3338, 1997.

[3] J.F. Cardoso. High-order constrast for independent component analysis. *Neural Computation*, 11:157–192, 1999.

[4] P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36:287–314, 1994.

[5] F.R. Hampel, E.M. Ronchetti, P.J. Rousseuw, and W.A. Stahel. *Robust statistics*. Wiley, 1986.

[6] P.J. Huber. *Robust statistics*. Wiley, 1981.

[7] A. Hyvärinen. New approximations of differential entropy for independent component analysis and projection pursuit. In *Advances in Neural Information Processing Systems*, volume 10, pages 273–279, 1998.

[8] M.G. Kendall and A. Stuart. *The advanced theory of statistics Vol. 1*. Griffin, 1963.

[9] P. McCullagh. *Tensor Methods in Statistics*. Chapman and Hall, 1987.

[10] M. Welling and M. Weber. A constrained EM algorithm for independent component analysis. *Neural Computation*, 13:677–689, 2001.

Online (and Offline) on an Even Tighter Budget

Jason Weston
NEC Laboratories America,
Princeton, NJ, USA
jasonw@nec-labs.com

Antoine Bordes
NEC Laboratories America,
Princeton, NJ, USA
antoine@nec-labs.com

Leon Bottou
NEC Laboratories America,
Princeton, NJ, USA
leonb@nec-labs.com

Abstract

We develop a fast online kernel algorithm for classification which can be viewed as an improvement over the one suggested by (Crammer, Kandola and Singer, 2004), titled "Online Classificaton on a Budget". In that previous work, the authors introduced an *on-the-fly* compression of the number of examples used in the prediction function using the size of the margin as a quality measure. Although displaying impressive results on relatively noise-free data we show how their algorithm is susceptible in noisy problems. Utilizing a new quality measure for an included example, namely the error induced on a selected subset of the training data, we gain improved compression rates and insensitivity to noise over the existing approach. Our method is also extendable to the batch training mode case.

1 Introduction

Rosenblatt's Perceptron (Rosenblatt, 1957) efficiently constructs a hyperplane separating labeled examples $(x_i, y_i) \in \mathbb{R}^n \times \{-1, +1\}$. Memory requirements are minimal because the Perceptron is an *online* algorithm: each iteration considers a single example and updates a candidate hyperplane accordingly. Yet it globally converges to a separating hyperplane if such a hyperplane exists.

The Perceptron returns an arbitrary separating hyperplane regardless of the minimal distance, or *margin*, between the hyperplane and the examples. In contrast, the Generalized Portrait algorithm (Vapnik and Lerner, 1963) explictly seeks an hyperplane with maximal margins.

All the above methods produce a hyperplane whose normal vector is expressed as a linear combination of examples. Both training and recognition can be carried out with the only knowledge of the dot products $x_i' x_j$ between examples. Support Vector Machines (Boser, Guyon and Vapnik, 1992) produce maximum margin non-linear separating hypersurfaces by simply replacing the dot products by a Mercer kernel $K(x_i, x_j)$.

Neither the Generalized Portrait nor the Support Vector Machines (SVM) are online algorithms. A set of training examples must be gathered (and stored in memory) prior to running the algorithm. Several authors have proposed online Perceptron variants that feature both the margin and kernel properties. Example of such algorithms include the Relaxed Online Maximum Margin Algorithm (ROMMA) (Li and Long, 2002), the Approximate Maximal Margin Classification Algorithms (ALMA) (Gentile, 2001), and the Margin Infused Relaxed Algorihm (MIRA) (Crammer and Singer, 2003).

The computational requirements[1] of kernel algorithms are closely related to the *sparsity* of the linear combination defining the separating hyper-surface. Each iteration of most Perceptron variants considers a single example and decides whether to insert it into the linear combination. The Budget Perceptron (Crammer, Kandola and Singer, 2004) achieves greater sparsity by also trying to remove some of the examples already present in the linear combination.

This discussion only applies to the case where all examples can be separated by a hyperplane or a hypersurface, that is to say in the absence of noise. Support Vector Machines use Soft Margins (Cortes and Vapnik, 1995) to handle noisy examples at the expense of sparsity. Even in the case where the training examples can be separated, using Soft Margins often improves the test error. Noisy data sharply degrades the performance of all the Perceptron variants discussed above.

We propose a variant of the Perceptron algorithm that addresses this problem by removing examples from the linear combination on the basis of a direct measurement of the training error in the spirit of Kernel Matching Pursuit (KMP) (Vincent and Bengio, 2000). We show that this algorithm has good performance on both noisy and non-noisy data.

[1] and sometimes the generalization properties

2 Learning on a Budget

Figure 1 shows the Budget Perceptron algorithm (Crammer, Kandola and Singer, 2004). Like Rosenblatt's Perceptron, this algorithm ensures that the hyperplane normal $\mathbf{w}_t$ can always be expressed as a linear combination of the examples in set C_t:

$$\mathbf{w}_t = \sum_{i \in C_t} \alpha_i \mathbf{x}_i. \quad (1)$$

Whereas Rosenblatt's Perceptron updates the hyperplane normal w_t whenever the current example $(\mathbf{x}_t, y_t)$ is misclassified, the Budget Perceptron updates the normal whenever the margin is smaller than a predefined parameter $\beta > 0$, that is to say whenever $y_t(\mathbf{x}_t \cdot \mathbf{w}_t) < \beta$.

Choosing a large β ensures that the hyperplane will eventually become close to the maximal margin hyperplane. This also increases the likelihood that an arbitrary example will become part of the expansion (1) and make the final solution less sparse.

The Budget Perceptron addresses this problem with a *removal* process. Whenever the number of expansion examples exceeds a predefined threshold p, the removal process excludes one example from the expansion. More specifically, the removal process (steps $1a$–$1c$, figure 1) simulates the removal of each example and eventually selects the example i that, when removed, remains recognized with the largest margin:

$$i = \arg\max_{j \in C_t} \{y_j(\mathbf{w}_{t-1} - \alpha_j y_j \mathbf{x}_j) \cdot \mathbf{x}_j\}$$

The justification for such a strategy is that the Perceptron algorithm only adds examples to the cache when they are errors. Early on in the training, examples may be added because the decision rule learnt thus far is relatively inaccurate, however later on these examples may be well classified as the direction of the hyperplane has changed considerably. The standard Perceptron algorithm does not have any removal procedure.

Several variants of this algorithms can be derived by changing the update formula (figure 2) or by replacing the dot products by suitable kernel functions. The maximum size of the expansion can be fixed or variable (Crammer, Kandola and Singer, 2004). Essentially, to adapt to the variable case one removes all examples that violate $y_j(w_{t-1} - \alpha_j y j x_j) \cdot x_j < \beta$ on each iteration. For simplicity however, in the remainder of the paper we will present algorithms in the simplest linear setup with Perceptron update and fixed sized cache, and leave such variants to the reader.

Experimental results (Crammer, Kandola and Singer, 2004) demonstrate that the Budget Perceptron performs extremely well on relatively noiseless problems. However, it degrades quickly on noisy problems. Suppose for instance

Input: Margin $\beta > 0$, Cache Limit p.

Initialize: Set $\forall t\ \alpha_t = 0$, $\mathbf{w}_0 = 0$, $C_0 = \emptyset$

Loop: For $t = 1, \ldots, T$

- Get a new instance $\mathbf{x}_t \in \mathbb{R}^n$, $y_t = \pm 1$.
- Predict $\hat{y}_t = \text{sign}(y_t(\mathbf{x}_t \cdot \mathbf{w}_{t-1}))$
- If $y_t(\mathbf{x}_t \cdot \mathbf{w}_{t-1}) \leq \beta$ update:
  1. If $|C_t| = p$ remove one example:
     a Find $i = \arg\max_{j \in C_t} \{y_j(\mathbf{w}_{t-1} - \alpha_j y_j \mathbf{x}_j) \cdot \mathbf{x}_j\}$
     b Update $\mathbf{w}_{t-1} \leftarrow \mathbf{w}_{t-1} - \alpha_i y_i \mathbf{x}_i$.
     c Remove $C_{t-1} \leftarrow C_{t-1} \setminus \{i\}$.
  2. Insert $C_t \leftarrow C_{t-1} \cup \{t\}$.
  3. Set $\alpha_t = 1$.
  4. Compute $\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} + \alpha_t y_t \mathbf{x}_t$.

Output: $H(x) = \text{sign}(\mathbf{w}_T \cdot \mathbf{x})$.

Figure 1: The Budget Perceptron algorithm (Crammer, Kandola and Singer, 2004).

that we randomly flip the labels of a small proportion η of both the training and test examples. The misclassification rate of the best hyperplane is at least η. Such misclassified examples accumulate into the Budget Perceptron expansion because only examples which are *classified well* are removed. Mislabeled examples reverse the direction of the normal w_t, and poor performance follows.

Complexity Assuming we use an RBF kernel, the insertion step requires $O(pn)$ operations where p is the cache size and n is the input dimensionality. The deletion step requires $O(p)$ operations, assuming all kernel calculations are cached. Note that the latter cost is only incurred for margin errors when the cache is full.

Perceptron (Rosenblatt, 1957)

$$\alpha_t = 1$$

MIRA (Crammer and Singer, 2003)

$$\alpha_t = \min\left(1, \max\left(0, \frac{-y_i(\mathbf{w} \cdot \mathbf{x}_i)}{\mathbf{x}_i \cdot \mathbf{x}_i}\right)\right)$$

No-Bias-SVM (Kecman, Vogt and Huang, 2003) $\beta = 1$

$$\alpha_t = \min\left(C, \max\left(0, \frac{1 - y_i(\mathbf{w} \cdot \mathbf{x}_i)}{\mathbf{x}_i \cdot \mathbf{x}_i}\right)\right)$$

Figure 2: **Update Rules for Various Algorithms.** These can be used to replace step 3 in figure 1 or 3.

3 Learning on a Tighter Budget

The Budget Perceptron removal process simulates the removal of each example and eventually selects the example that remains recognized with the largest margin. This margin can be viewed as an indirect estimate of the impact of the removal on the overall performance of the hyperplane. Thus, to improve the Budget algorithm we propose to replace this indirect estimate by a direct evaluation of the misclassification rate. We term this algorithm the Tighter Budget Perceptron. The idea is simply to replace the measure of margin

$$i = \arg\max_{j \in C_t} \{ y_j (\mathbf{w}_{t-1} - \alpha_j y_j \mathbf{x}_j) \cdot \mathbf{x}_j \} \qquad (2)$$

with the overall error rate on all currently seen examples:

$$i = \arg\min_{j \in C_t} \{ \sum_{k=1}^{t} L(y_k, \text{sign}((\mathbf{w}_{t-1} - \alpha_j y_j \mathbf{x}_j) \cdot \mathbf{x}_k) \}.$$

Intuitively, if an example is well classified (has a large margin) then not only will it be correctly classified when it is removed as in equation (2) but also all other examples will still be well classified as well. On the other hand, if an example is an outlier then its contribution to the expansion of w is likely to classify points incorrectly. Therefore when removing an example from the cache one is likely to either remove noise points or well-classified points first. Apart from when the kernel matrix has very low rank we do indeed observe this behaviour, e.g. in figure 8.

Compared to the original Budget Perceptron, this removal rule is more expensive to compute, now it requires $O(t(p+n))$ operations (see section 2). Therefore in section 3.2 we discuss ways of approximating this computation whilst still retaining its desirable properties. First, however we discuss the relationship between this algorithm and existing approaches.

3.1 Relation to Other Algorithms

Kernel Matching Pursuit The idea of kernel matching pursuit (KMP) (Vincent and Bengio, 2000) is to build a predictor $w = \sum_i \alpha_i x_i$ greedily by adding one example at a time, until a pre-chosen cache size p is found. The example to add is chosen by searching for the example which gives the largest decrease in error rate, usually in terms of squared loss, but other choices of loss function are possible. While this procedure is for *batch* learning, and not *online* learning, clearly this criteria for *addition* is the same as our criteria for *deletion*.

There are various variants of KMP, two of them called basic- and backfitting- are described in figure 4. Basic adapts only a single α_i in the insertion step, whereas backfitting adjusts all α_i of previously chosen points. The latter

Input: Margin $\beta > 0$, Cache Limit p.

Initialize: Set $\forall t\ \alpha_t = 0$, $\mathbf{w}_0 = 0$, $C_0 = \emptyset$.

Loop: For $t = 1, \ldots, T$

- Get a new instance $\mathbf{x}_t \in \mathbb{R}^n$, $y_t = \pm 1$.
- Predict $\hat{y}_t = \text{sign}(y_t(\mathbf{x}_t \cdot \mathbf{w}_{t-1}))$.
- Get a new label y_t.
- If $y_t(\mathbf{x}_t \cdot \mathbf{w}_{t-1}) \leq \beta$ update:
  1. If $|C_t| = p$ remove one example:
     a Find $i = \arg\min_{j \in C_t}$ $\{\sum_{k=1}^{t} L(y_k, \text{sign}((\mathbf{w}_{t-1} - \alpha_j y_j \mathbf{x}_j) \cdot \mathbf{x}_k)\}$.
     b Update $\mathbf{w}_{t-1} \leftarrow \mathbf{w}_{t-1} - \alpha_i y_i \mathbf{x}_i$
     c Remove $C_{t-1} \leftarrow C_{t-1} \setminus \{i\}$.
  2. Insert $C_t \leftarrow C_{t-1} \cup \{t\}$.
  3. Set $\alpha_t = 1$.
  4. Compute $\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} + \alpha_t y_t \mathbf{x}_t$.

Output: $H(x) = \text{sign}(\mathbf{w}_T \cdot \mathbf{x})$.

Figure 3: The Tighter Budget Perceptron algorithm.

can be computed efficiently if the kernel matrix can be fit into memory (the algorithm is given in (Vincent and Bengio, 2000)), but is expensive for large datasets. The basic algorithm, on the other hand does not perform well for classification, as shown in figure 8.

Note that we could adapt our algorithm's addition step to also be based on training error. However, using the Perceptron rule, an example is only added to the cache if it is an error, making it more efficient to compute. Note that variants of KMP have also been introduced that incorporate a deletion as well as an insertion step (Nair, Choudhury and Keane, 2002).

Condense and Multi-edit Condense and multi-edit (Devijver and Kittler, 1982) are editing algorithms to "sparsify" k-NN. Condense removes examples that are far from the decision boundary. The Perceptron and the SVM already have their own "condense" step as such points typically have $\alpha_i = 0$. The Budget Perceptron is an attempt to make the condense step of the Perceptron more aggressive. Multi-edit attempts to remove all the examples that are on the wrong side of the Bayes decision boundary. One is then left with learning a decision rule with non-overlapping classes with the same Bayes decision boundary as before, but with Bayes risk equal to zero. Note that neither the Perceptron nor the SVM (with soft margin) perform this step[2], and all incorrectly classified examples become support vec-

[2] An algorithm designed to combine the multi-edit step into SVMs is developed in (Bakır, Bottou and Weston, 2004).

Input: Cache Limit p.

Initialize: Set $\forall t\ \alpha_t = 0$, $\mathbf{w}_0 = 0$, $C_0 = \emptyset$.

Loop: For $t = 1, \ldots, p$

- Choose $(k, \alpha) =$
$$\arg\min_{\alpha,\ j=1\ldots m} \sum_{i=1}^{m} (y_j - (\mathbf{w}_t + \alpha \mathbf{x}_j) \cdot x_i)^2.$$
- Insert $C_t \leftarrow C_{t-1} \cup \{t\}$.
- *Basic-KMP:*
  Set $\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} + \alpha \mathbf{x}_k$.
  Backfitting-KMP:
  Set $\mathbf{w}_t \leftarrow \sum_{i \in C_t} \alpha_i \mathbf{x}_i$ where $\{\alpha_i\} =$
$$\arg\min_{\{\alpha_i\}} \sum_{j=1}^{m} \left(y_j - \sum_{i \in C_t} \alpha_i \mathbf{x}_i \cdot \mathbf{x}_j) \right)^2$$

Output: $H(x) = \text{sign}(\mathbf{w}_T \cdot \mathbf{x})$.

Figure 4: The Basic and Backfitting Kernel Matching Pursuit (KMP) Algorithms (Vincent and Bengio, 2000).

tors with $\alpha_i > 0$. Combining condense and multi-edit together one only tries to keep the correctly classified examples close to the decision boundary. The Tighter Budget Perceptron is also an approximate way of trying to achieve these two goals, as previously discussed.

Regularization One could also view the Tighter Budget Perceptron as an approximation of minimizing a regularized objective of the form

$$\frac{1}{m} \sum L(y_i, f(x_i)) + \gamma ||\alpha||_0.$$

where operator $|| \cdot ||_0$ is defined as counting the number of nonzero coefficients. That is to say, the fixed sized cache chosen acts a regularizer to reduce the capacity of the set of functions implementable by the Perceptron rule, the goal of which is to minimize the classification loss. This means that for noisy problems, with a reduced cache size one should see improved generalization error compared to a standard Perceptron using the Tighter Budget Perceptron, and we indeed find experimentally that this is the case.

3.2 Making the per-time-step complexity bounded by a constant independent of t

An important requirement of online algorithms is that their per-time-step complexity should be bounded by a constant independent of t (t being the time-step index), for it is assumed that samples arrive at a constant rate. The algorithm in figure 3 grows linearly in the time, t, because of the computation in step 1(a), that is when we choose the example

Input: Q_{t-1}, $\mathbf{x}_t$, $\mathbf{s}$

- $Q_t \leftarrow Q_{t-1} \cup \mathbf{x}_t$.
- If $|Q_t| > q$
  1. $i = \arg\max_{i \in Q_{t-1}} \mathbf{s}_i$
  2. $Q_t \leftarrow Q_t \setminus \mathbf{x}_i$

Output: Q_t

Figure 5: Algorithm for maintaining a fixed cache size q of relevant examples for estimating the training error. The idea is to maintain a count s_i of the number of times the prediction changes label for example i. One then retains the examples which change labels most often.

in the cache to delete which results in the minimal loss over all t observations:

$$i = \arg\min_{j \in C_t} \{ \sum_{k=1}^{t} L(y_k, \text{sign}((\mathbf{w}_{t-1} - \alpha_j y_j \mathbf{x}_j) \cdot \mathbf{x}_k) \}.$$

(Note that this extra computational expense is only invoked when $\mathbf{x}_t$ is a margin error, which if the problem has a low error rate, is only on a small fraction of the iterations.) Nevertheless, it is possible to consider approximations to this equation to make the algorithm independent of t.

We could simply reduce the measure of loss to only the fixed p examples in the cache:

$$i = \arg\min_{j \in C_t} \{ \sum_{k \in C_t} L(y_k, \text{sign}((\mathbf{w}_{t-1} - \alpha_j y_j \mathbf{x}_j) \cdot \mathbf{x}_k) \}. \quad (3)$$

While this is faster to compute, it may be suboptimal as we wish to have an estimator of the loss that is as unbiased as possible, and the points that are selected in the cache are a biased sample. However, they do have the advantage that many of them may be close to the decision surface.

A more unbiased sample could be chosen simply by picking a fixed number of randomly chosen examples, say q examples, where we choose q in advance. We define this subset as Q_t, where $|Q_t| = \min(q, t)$ which is taken from the t available examples until the cache is filled. Then we compute:

$$i = \arg\min_{j \in C_t} \{ \sum_{k \in Q_t} L(y_k, \text{sign}((\mathbf{w}_{t-1} - \alpha_j y_j \mathbf{x}_j) \cdot \mathbf{x}_k) \}. \quad (4)$$

The problem with this strategy is that many of these examples may be either very easy to classify or always mislabeled (noise) so this could be wastful.

We therefore suggest a secondary caching scheme to choose the q examples with which we estimate the error. We wish to keep the examples that are most likely to change

label as these are most likely to give us information about the performance of the classifier. If an example is well classified it will not change label easily when the classifier changes slightly. Likewise, if an example is an outlier it will be consisently incorrectly classified. In fact the number of examples that are relevant in this context should be relatively small. We therefore keep a count $\mathbf{s}_i$ of the number of times example $\mathbf{x}_i$ has changed label, divided by the amount of time it has been in the cache. If this value is small then we can consider removing this point from the secondary cache. When we receive a new observation at $\mathbf{x}_t$ at time t we thus perform the update given in figure 5 in the case that $\mathbf{x}_t$ is a margin error.

Complexity The last variant of the Tighter Budget Perceptron has a deletion step cost of $O(pq + qn)$ operations, where p is the cache size, q is the secondary cache size, and n is the input dimensionality. This should be compared to $O(p)$ for the Budget Perceptron, where clearly the deletion step is still less expensive.

In the case of relatively noise free problems with a reasonable cache size p, the deletion step occurs infrequently: by the time the cache becomes full, the perceptron performs well enough to make margin errors rare. The insertion step then dominates the computational cost. In the case of noisy problems, the cheaper deletion step of the Budget Perceptron performs too poorly to be considered a valid alternative. Moreover, as we shall see experimentally, the Tighter Budget Perceptron can achieve the same test error as the Budget Perceptron for smaller cache size p.

4 Experiments

4.1 2D Experiments - Online mode

Figure 6 shows a 2D classification problem of 1000 points from two classes separated by a very small margin. We show the decision rule found after one epoch of Perceptron, Budget Perceptron and Tighter Budget Perceptron training, using a linear kernel. Both Budget Perceptrons variants produce sparser solutions than the Perceptron, although the Budget Perceptron provides slightly less accurate solutions, even for larger cache sizes. Figure 7 shows a similar dataset, but with overlapping classes. The Perceptron algorithm will fail to converge with multiple epochs in this case. After one epoch a decision rule is obtained with a relatively large number of SVs. Most examples which are on the wrong side of the Bayes decision rule are SVs. [3] The Budget Perceptron fails to alleviate this problem. Although one can reduce the cache size to force more sparsity, the decision rule obtained is highly inaccurate. This is due to noise points which are far from the decision boundary being the last vectors to be removed from the cache, as can be seen in the example with a cache size of 50.

[3] Note that support vector machines, not shown, also suffer from a similar deficiency in terms of sparsity - all incorrectly classified examples are SVs.

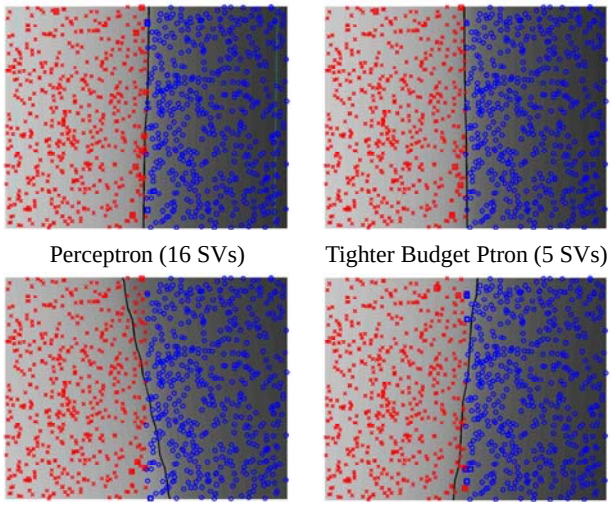


Figure 6: **Separable Toy Data in Online Mode.** The Budget Perceptron of (Crammer, Kandola and Singer, 2004) and our Tighter Budget Perceptron provide sparser solutions than the Perceptron algorithm, however the Budget Perceptron seems sometimes to provide slightly worse solutions.

4.2 2D Experiments - Batch mode

Figure 8 shows a 2D binary classification problem with the decision surface found by the Tighter Budget Perceptron, Budget Perceptron, Perceptron, SVM, and two flavors of KMP when using the same Gaussian kernel. For the online algorithms we ran multiple epochs over the dataset until convergence. This example gives a simple demonstration of how the Budget and Tighter Budget Perceptrons can achieve a greater level of sparsity than the Perceptron, whilst choosing examples that are close to the margin, in constrast to the KMP algorithm.

Where possible in the fixed cache algorithms, we fixed the cache sizes to 10 SVs (examples highlighted with squares), as a trained SVM uses this number. The Perceptron is not as sparse as SVM, and uses 19 SVs. However both the Budget Perceptron and the Tighter Budget Perceptron still separate the data with 10 SVs. The Perceptron required 14 epochs to converge, the Tighter Budget Perceptron required 22, the Budget Perceptron required 26 (however, we had to decrease the width of the Gaussian kernel for the last algorithm as it did not converge for larger widths). Backfitting-KMP provides as good sparsity as SVM. Basic-KMP does not give zero error even after 400 iterations, and by this time has used 37 SVs (it cycles around the same SVs many times). Note that all the algorithms except KMP choose

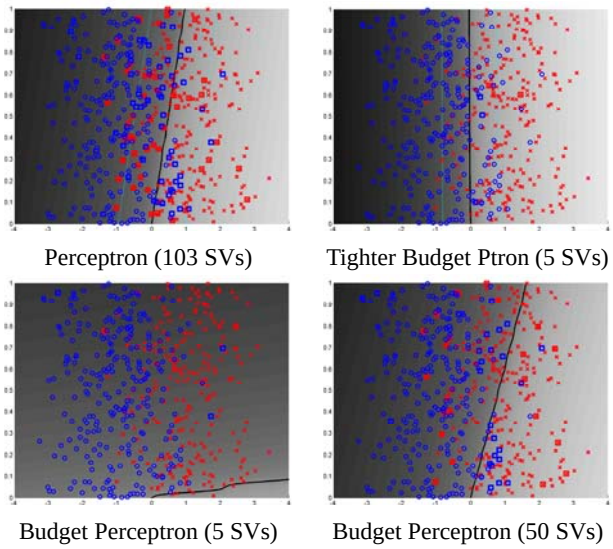


Figure 7: **Noisy Toy Data in Online Mode.** The Perceptron and Budget Perceptron (independent of cache size) fail when problems contain noise, as demonstrated by a simple 2D problem with Gaussian noise in one dimension. The Tighter Budget Perceptron, however, finds a separation very close to the Bayes rule.

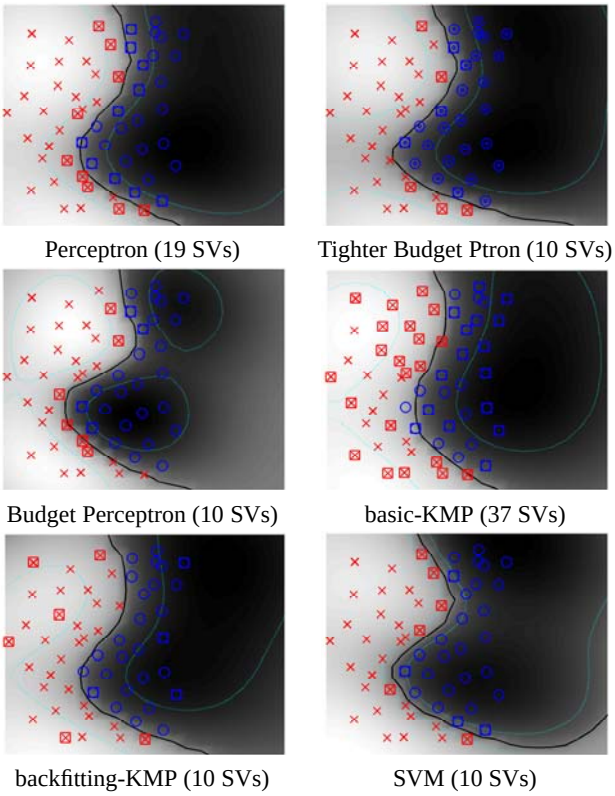


Figure 8: **Nonlinear Toy Data in Batch Mode.** We compare various algorithms on a simple nonlinear dataset following (Vincent and Bengio, 2000). See the text for more explanation.

SVs close to the margin.

4.3 Benchmark Datasets

We conducted experiments on three well-known datasets: the US Postal Service (USPS) Database, Waveform and Banana[4]. A summary of the datasets is given in Table 1. For all three cases (USPS,Waveform,Banana) we chose an RBF kernel, the width values were taken from (Vapnik, 1998) and (Rätsch, Onoda and Müller, 2001), and are chosen to be optimal for SVMs, the latter two by cross validation. We used the same widths for all other algorithms, despite that these may be suboptimal in these cases. For USPS we considered the two class problem of separating digit zero from the rest. We also constructed a second experiment by corrupting USPS with label noise. We randomly flipped the labels of 10% of the data in the training set to observe the performance effects on the algorithms tested (for SVMs, we report the test error with the optimal value of C chosen on the test set, in this case, $C = 10$).

We tested the Tighter Budget Perceptron, Budget Perceptron and Perceptron in an *online* setting by only allowing one pass through the training data. Obviously this puts these algorithms at a disadvantage compared to batch algorithms such as SVMs which can look at training points multiple times. Nevertheless, we compare with SVMs as our gold standard measure of performance. The results are given in figures 9-12. In all cases for the Perceptron variants we use $\beta = 0$ and for the Tighter Budget Perceptron we employ the algorithm given in figure 3 without the computational efficiency techniques given in section 3.2. We show the test error against different fixed cache sizes p resulting in p support vectors. We report averages over 5 runs for USPS and 10 runs for Waveform and Banana. The error bars indicate the maximum and minimum values over all runs.

The results show the Tighter Budget Perceptron yielding similar test error performance to the SVM but with considerably less SVs. The Budget Perceptron fares less well with

Name	Inputs	Train	Test	σ	C
USPS	256	7329	2000	128	1000
Waveform	21	4000	1000	3.16	1
Banana	2	4000	1300	0.7	316

Table 1: **Datasets used in the experiments.** The hyperparameters are for an SVM with RBF kernel, taken from (Vapnik, 1998) and (Rätsch, Onoda and Müller, 2001).

[4]USPS is available at `ftp://ftp.kyb.tuebingen.mpg.de/pub/bs/data`. Waveform and Banana are available at `http://mlg.anu.edu.au/~raetsch/data/`.

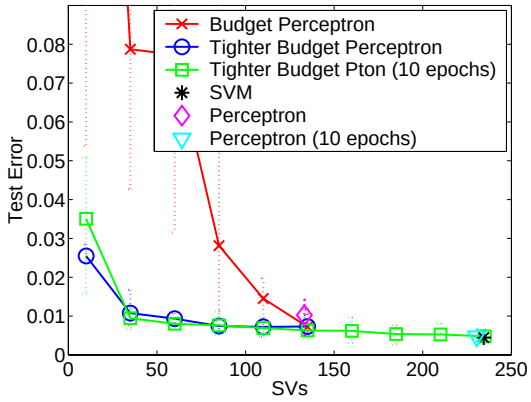


Figure 9: **USPS Digit 0 vs Rest.**

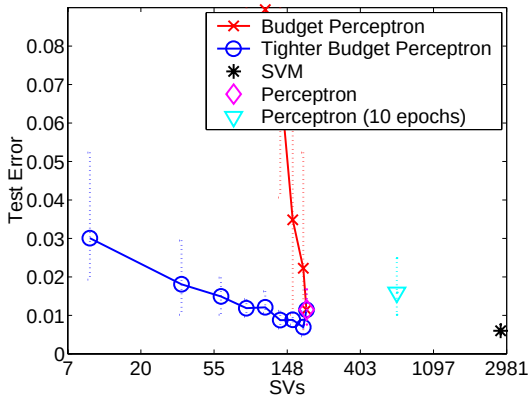


Figure 10: **USPS Digit 0 vs Rest + 10% noise.**

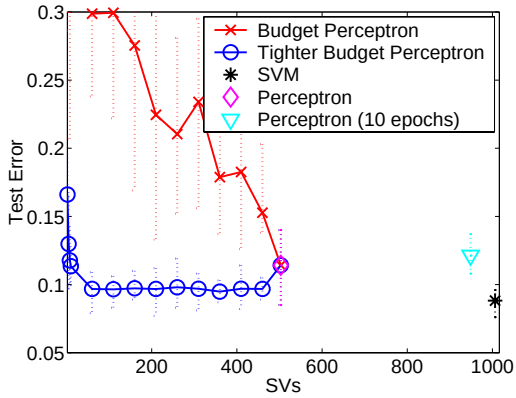


Figure 11: **Waveform Dataset.**

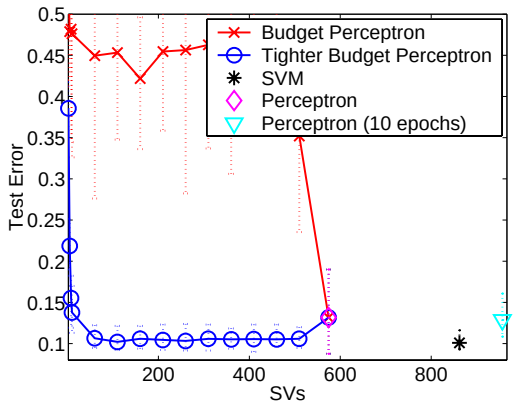


Figure 12: **Banana Dataset.**

the test error degrading considerably faster for decreasing cache size compared to the Tighter Budget Perceptron, particularly on the noisy problems. Note that if the cache size is large enough both the Budget and Tighter Budget Perceptrons converge to the Perceptron solution, hence the two curves meet at their furthest point. However, while the test error immediately degrades for the Budget Perceptron, for the Tighter Budget Perceptron the test error in fact *improves* over the Perceptron test error in both the noisy problems. This should be expected as the standard Perceptron cannot deal with overlapping classes.

In figure 9 we also show the Tighter Budget Perceptron with (up to) 10 epochs on USPS (typically the algorithm converges before 10 epochs). The performance is similar to only 1 epoch for small cache sizes. For larger cache sizes, clearly the maximum cache size converges to the same performance of a Perceptron with 10 epochs, which in this case gives slightly better performance than any cache size possible with 1 epoch.

4.4 Faster Error Computation

In this section we explore the error evaluation cache strategies described previously in section 3.2. We compared the following strategies to evaluate error rates: (i) using all points so far seen, (ii) using only the support vectors in the cache, i.e. equation (3), (iii) using a random cache of size q, i.e. equation (4), and (iv) using a cache of size q of the examples that flip label most often, i.e. figure 5.

Figure 13 compares these methods on the USPS dataset for fixed cache size of support vectors $p = 35$ and $p = 85$. The results are averaged over 40 runs (the error bars show the standard error).

We compare different amounts of evaluation vectors q. The results show that considerable computational speedup can be gained by any of the methods compared to keeping all training vectors. Keeping a cache of examples that change label most often performs better than a random cache for small cache sizes. Using the support vectors themselves also performs better than the random strategy for the same cache size. This makes sense as support vectors themselves are likely to be examples that can change label easily, making it similar to the cache of examples that most often change label. Nevertheless, it can be worthwhile to have a small number of support vectors for fast evaluation, but a larger set of error evaluation vectors when an error is encountered. We suggest to choose q and p such that a cache of the qp kernel calculations fits in memory at all times.

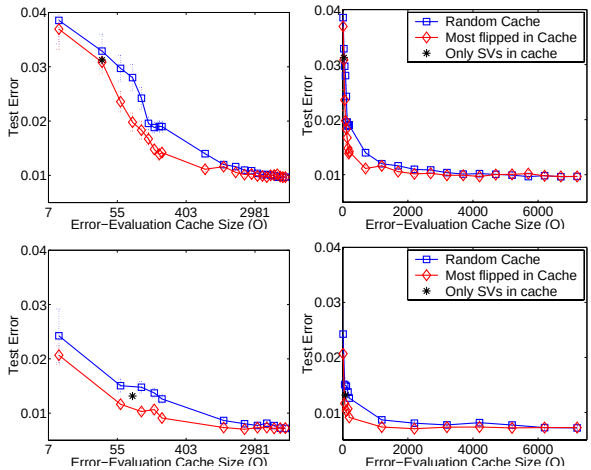


Figure 13: **Error Rates for Different Error Measure Cache Strategies on USPS, digit zero versus the rest.** The number of SVs is fixed to 35 in the top row, and 85 in the bottom row, the left-hand plots are log plots of the right-hand ones. The different strategies change the number of points used to evaluate the error rate for the SV cache deletion process.

5 Summary

We have introduced a sparse online algorithm that is a variant of the Perceptron. It attempts to deal with some of the computational issues of using kernel algorithms in an online setting by restricting the number of SVs one can use. It also allows methods such as the Perceptron to deal with overlapping classes and noise. It can be considered as an improvement over the Budget Perceptron of (Crammer, Kandola and Singer, 2004) because it is empirically sparser than that method for the same error rate, and can handles noisy problems while that method cannot. Our method tends to keep only points that are close to the margin *and* that lie on the correct side of the Bayes decision rule. This occurs because other examples are less useful for describing a decision rule with low error rate, which is the quality measure we use for inclusion in to the cache.

However, the cost of this is that quality measure used to evaluate training points is more expensive to compute than for the Budget Perceptron (and in this sense the name "Tighter Budget Perceptron" is slightly misleading). However, we believe there exist various approximations to speed up this method whilst retaining its useful properties. We explored some strategies in this vein by introducing a small secondary cache of evaluation vectors with positive results. Future work should investigate further ways to improve on this, some first suggestions being to only look at a subset of points to remove on each iteration, or to remove the worst n points every n iterations.

Acknowledgements

Part of this work was funded by NSF grant CCR-0325463.

References

Bakır, G., Bottou, L., and Weston, J. (2004). Breaking SVM Complexity with Cross-Training. In *Advances in Neural Information Processing Systems 17 (NIPS 2004)*. MIT Press, Cambridge, MA. to appear.

Boser, B. E., Guyon, I. M., and Vapnik, V. (1992). A Training Algorithm for Optimal Margin Classifiers. In Haussler, D., editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA. ACM Press.

Cortes, C. and Vapnik, V. (1995). Support Vector Networks. *Machine Learning*, 20:273–297.

Crammer, K., Kandola, J., and Singer, Y. (2004). Online Classification on a Budget. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

Crammer, K. and Singer, Y. (2003). Ultraconservative Online Algorithms for Multiclass Problems. *Journal of Machine Learning Research*, 3:951–991.

Devijver, P. and Kittler, J. (1982). *Pattern Recogniton, A statistical approach*. Prentice Hall, Englewood Cliffs.

Gentile, C. (2001). A New Approximate Maximal Margin Classification Algorithm. *Journal of Machine Learning Research*, 2:213–242.

Kecman, V., Vogt, M., and Huang, T. (2003). On the Equality of Kernel AdaTron and Sequential Minimal Optimization in Classification and Regression Tasks and Alike Algorithms for Kernel Machines. In *Proceedings of European Symposium on Artificial Neural Networks, ESANN'2003*, pages 215–222, Evere, Belgium. D-side Publications.

Li, Y. and Long, P. (2002). The Relaxed Online Maximum Margin Algorithm. *Machine Learning*, 46:361–387.

Nair, P. B., Choudhury, A., and Keane, A. J. (2002). Some Greedy Learning Algorithms for Sparse Regression and Classification with Mercer Kernels. *Journal of Machine Learning Research*, 3:781–801.

Rätsch, G., Onoda, T., and Müller, K.-R. (2001). Soft Margins for AdaBoost. *Machine Learning*, 42(3):287–320. Also: NeuroCOLT Technical Report 1998-021.

Rosenblatt, F. (1957). The Perceptron: A perceiving and recognizing automaton. Technical Report 85-460-1, Project PARA, Cornell Aeronautical Lab.

Vapnik, V. and Lerner, A. (1963). Pattern Recognition using Generalized Portrait Method. *Automation and Remote Control*, 24:774–780.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley, New York.

Vincent, P. and Bengio, Y. (2000). Kernel Matching Pursuit. Technical Report 1179, Département d'Informatique et Recherche Opérationnelle, Université de Montréal. Presented at Snowbird'00.

Approximations with Reweighted Generalized Belief Propagation

Wim Wiegerinck
SNN, Radboud University Nijmegen
Geert Grooteplein 21, 6525 EZ, Nijmegen, The Netherlands
W.Wiegerinck@science.ru.nl

Abstract

In (Wainwright et al., 2002) a new general class of upper bounds on the log partition function of arbitrary undirected graphical models has been developed. This bound is constructed by taking convex combinations of tractable distributions. The experimental results published so far concentrates on combinations of tree-structured distributions leading to a convexified Bethe free energy, which is minimized by the tree-reweighted belief propagation algorithm. One of the favorable properties of this class of approximations is that increasing the complexity of the approximation is guaranteed to increase the precision. The lack of this guarantee is notorious in standard generalized belief propagation. We increase the complexity of the approximating distributions by taking combinations of junction trees, leading to a convexified Kikuchi free energy, which is minimized by reweighted generalized belief propagation. Experimental results for Ising grids as well as for fully connected Ising models are presented illustrating advantages and disadvantages of the reweighting method in approximate inference.

1 INTRODUCTION

Probabilistic graphical models such as Bayesian networks and Markov random fields are powerful tools for learning and reasoning in domains with uncertainty. Unfortunately, exact inference is intractable in large, complex graphs. Therefore approximate inference methods are of great importance. An approximation method that recently received much attention is loopy belief propagation (BP) (Pearl, 1988). Although the algorithm is not guaranteed to converge, it often gives surprisingly accurate results (Murphy et al., 1999). In (Yedidia et al., 2001), it has been shown that fixed points of loopy BP are actually extrema of the Bethe free energy, which can be considered as a two-node approximation of the exact free energy of the system. By considering the Kikuchi free energy, which is an approximate free energy based on larger clusters of nodes, the more advanced generalized belief propagation (GBP) algorithm has been derived (Yedidia et al., 2001). This algorithm can be viewed as an interpolation between loopy BP and the junction tree algorithm (Lauritzen and Spiegelhalter, 1988; Jensen, 1996). The relation between (G)BP and the approximate Bethe/Kikuchi free energies motivated several researchers to design double-loop algorithms for explicit minimization of the Bethe/Kikuchi free energy with a guaranteed convergence to a (local) optimum (Teh and Welling, 2002; Yuille, 2002; Heskes et al., 2003).

Increasing the cluster-size in GBP often improves the accuracy of the approximation. Unfortunately, this is not guaranteed. A notorious counter example is the fully connected Ising model with pair and triplet approximations (Kappen and Wiegerinck, 2002). Even with moderate interaction strength, the triplet approximation is much worse than the pair approximation. A related problem is that the quality of different GBP approximations cannot be compared by comparing their Kikuchi free energy values. A lower Kikuchi free energy does not imply a better approximation.

A method that is closely related to (G)BP, but derived in a completely different way is the convexified free energy approximation (Wainwright et al., 2002; Wainwright et al., 2003; Wainwright and Jordan, 2003). This approximation is derived to provide an upper bound of the log partition function. The parameters of the exact model are represented as the average of parameters of tractable models. By the convexity of the log partition function, it is upper-bounded by the average of the log partition functions of the tractable models. The optimization of this upper bound then

give rise to a approximate Bethe/Kikuchi-like free energy, in which the cluster beliefs are parameters to be optimized. The advantage of this approach is that this approximate free energy is convex. So it is relatively easy to minimize, and it is guaranteed to have only a single minimum. Another advantage of this method is that a nested sequence of approximations with increasing complexity leads to tighter approximations of the free energy. So, for example, in a fully connected Ising model, the convexified approximation using triplets must be more accurate in free energy than with using pairs. One may expect (or hope) that the increase of precision in free energy is reflected in an increase of precision of other quantities, such as node marginals. Experimental results published so-far only involved approximations with trees, leading to a convexified Bethe free energy, which is optimized by tree-reweighted BP. The optimized pseudo-marginals in these approximations are pair-marginals.

The main contribution of this paper is an experimental study of convexified approximations with increasingly complex clusters. First, we review the convexified free energy approximation of an arbitrary discrete probability distributions (Wainwright et al., 2002; Wainwright et al., 2003), and the use of convex combinations of junction-trees leading to convexified Kikuchi free energy, as outlined earlier in (Wainwright and Jordan, 2003). We present reweighted GBP (i.e., RGBP) to minimize the convexified free energy. This algorithm is a straightforward generalization of the message-free GBP algorithm presented in (Heskes and Zoeter, 2003). We consider Ising grids and fully connected Ising models for which we construct nested convexified pair and cluster approximations of the free energy and of the cluster marginals. These approximations are experimentally compared with each other and with the corresponding standard Bethe/Kikuchi approximations optimized with convergent double loop algorithms (Heskes et al., 2003).

2 EXACT MODEL, PARTITION FUNCTION AND FREE ENERGY

We consider a distribution over discrete variables $\mathbf{x}$ with potential $\psi(\mathbf{x})$,

$$P(\mathbf{x}) = \frac{1}{Z} \exp(\psi(\mathbf{x})) \,. \tag{1}$$

The normalization constant,

$$Z(\psi) = \sum_{\mathbf{x}} \exp(\psi(\mathbf{x})) \tag{2}$$

is known as the partition function. For later reference, we also define the variational free energy of the system,

$$F(\hat{P}) = -\sum_{\mathbf{x}} \hat{P}(\mathbf{x})\psi(\mathbf{x}) + \sum_{\mathbf{x}} \hat{P}(\mathbf{x}) \log \hat{P}(\mathbf{x}), \tag{3}$$

Minimizing the free energy with respect to $\hat{P}$ returns the distribution P in (1). The value of the free energy at its minimum is

$$F(P) = -\log Z \,. \tag{4}$$

2.1 JUNCTION TREES

Now we consider distributions over sets of nodes, $\mathbf{x} = (x_1, \ldots, x_n)$, and potentials that factorize into overlapping cluster potentials,

$$\psi(\mathbf{x}) = \sum_{\alpha} \psi_\alpha(\mathbf{x}_\alpha) \tag{5}$$

with $\alpha \subset \{1, \ldots, n\}$, and $\mathbf{x}_\alpha$ the state vector restricted to the the variables in α. The clusters and potentials are not uniquely defined. For instance, clusters can be merged into bigger clusters $\alpha'' = \alpha \cup \alpha'$ with $\psi_{\alpha''} = \psi_\alpha + \psi_{\alpha'}$. For convenience, we assume that clusters α are not contained in each other. This can be achieved by merging subclusters with their supersets.

In general the distribution $P(\mathbf{x})$ will be intractable. An exception is formed by models in which the (possibly merged) clusters can be organized into a junction tree with small maximal cluster size.

A junction tree (Lauritzen and Spiegelhalter, 1988; Jensen, 1996) is a hyper-tree of clusters $\alpha \in C$ (or actually: a forest of hyper-trees), which has the properties that for any pair of clusters α and α' with nonempty overlap $\alpha \cap \alpha'$: (1) There is a path in the cluster tree that connects α and α' and (2) All the clusters κ in the path connecting α and α' should contain their intersection: $\alpha \cap \alpha' \subset \kappa$ (running intersection property). The links between adjacent clusters in the hyper-tree are labeled with separators. They consists of the intersections $\delta = \alpha \cap \alpha'$ of the adjacent clusters. The number of times that a subcluster δ appears in the hyper-tree is n_δ.

A probabilistic model in which the cluster set $C = \{\alpha\}$ can be organized as a junction tree, can be factorized into a product of probabilities on the cliques C and separators S,

$$P(\mathbf{x}) = \frac{\prod_{\alpha \in \mathcal{C}} P(\mathbf{x}_\alpha)}{\prod_{\delta \in \mathcal{S}} P(\mathbf{x}_\delta)^{n_\delta}} \equiv \prod_{\gamma \in \Gamma} P(\mathbf{x}_\gamma)^{k_\gamma} \tag{6}$$

where we defined $\Gamma = C \cup S$, and the counting numbers $k_\gamma = 1$ for $\gamma \in C$ and $k_\gamma = -n_\gamma$ for $\gamma \in S$. The free

energy can be expressed as

$$F(P) = -\sum_{\alpha\in\mathcal{C}}\sum_{\mathbf{x}_\alpha}\psi_\alpha(\mathbf{x}_\alpha)P(\mathbf{x}_\alpha) + \sum_{\gamma\in\Gamma}k_\gamma\sum_{\mathbf{x}_\gamma}P(\mathbf{x}_\gamma)\log P(\mathbf{x}_\gamma) \quad (7)$$

By defining $P_\gamma(\mathbf{x}_\gamma) \equiv P(\mathbf{x}_\gamma)$, $\gamma \in \Gamma$ the minimization with respect to P can be conveniently be reformulated as a minimization with respect to the independent cluster marginals $\{P_\gamma(\mathbf{x}_\gamma)\}$ under constraint that they are consistent on their overlaps $\gamma \cap \gamma'$.

3 AN UPPER BOUND OF THE PARTITION FUNCTION

From now on, we assume that $P(\mathbf{x})$ is intractable. In this section the goal is to upper-bound $\log Z(\psi)$.

If we have a set of (tractable) distributions $P^{(\mathcal{T})}(\mathbf{x})$, with potentials $\phi^{(\mathcal{T})}(\mathbf{x})$, and a weight $\mu(\mathcal{T}) \geq 0$ for each of of the distributions, with $\sum_\mathcal{T}\mu(\mathcal{T}) = 1$, such that the original potential ψ is the weighted sum of potentials $\phi^{(\mathcal{T})}$

$$\sum_\mathcal{T}\mu(\mathcal{T})\phi^{(\mathcal{T})}(\mathbf{x}) = \psi(\mathbf{x}) \quad (8)$$

then the log partition function $\log Z(\psi)$ is upper bounded by

$$\log Z(\psi) = \log Z(\sum_\mathcal{T}\mu(\mathcal{T})\phi^{(\mathcal{T})}) \quad (9)$$

$$\leq \sum_\mathcal{T}\mu(\mathcal{T})\log Z(\phi^{(\mathcal{T})}) \quad (10)$$

This results from the semi-convexity of $\log Z(\psi)$, which follows from the fact the second derivative,

$$\frac{\partial^2 Z(\psi)}{\partial\psi(\mathbf{x})\partial\psi(\mathbf{y})} = P(\mathbf{x},\mathbf{y}) - P(\mathbf{x})P(\mathbf{y}) = \sum_\mathbf{z}P(\mathbf{z})(\delta_{\mathbf{zx}} - P(\mathbf{x}))(\delta_{\mathbf{zy}} - P(\mathbf{y})) \quad (11)$$

is positive semi-definite .

3.1 A CONVEX COMBINATION OF JUNCTION TREES

Now we take $P^{(\mathcal{T})}$ to be junction trees with cluster sets $C(\mathcal{T})$ (such that the maximal cluster size is small enough) and cluster potentials $\phi_\beta^{(\mathcal{T})}$. According to (8) these potentials should satisfy

$$\sum_\mathcal{T}\mu(\mathcal{T})\sum_{\beta\in C(\mathcal{T})}\phi_\beta^{(\mathcal{T})}(\mathbf{x}_\beta) = \psi(\mathbf{x}) \quad (12)$$

For convenience, we assume that each cluster α is a member of at least one cluster set $C(\mathcal{T})$, and that the clusters α are not smaller than the clusters β of the junction trees (i.e., $\alpha \subseteq \beta \Rightarrow \alpha = \beta$). This can be achieved by merging of clusters α.

To optimize the upper bound of the log partition function with respect to the cluster potentials $\{\phi_\beta^{(\mathcal{T})}\}$ for fixed $\{\mu(\mathcal{T})\}$, we construct the Lagrangian

$$\mathcal{L}(\{\phi^{(\mathcal{T})}\}, Q) = \sum_\mathcal{T}\mu(\mathcal{T})\log Z(\phi^{(\mathcal{T})}) + \sum_\mathbf{x}Q(\mathbf{x})\left[\psi(\mathbf{x}) - \sum_\mathcal{T}\mu(\mathcal{T})\sum_{\beta\in C(\mathcal{T})}\phi_\beta^{(\mathcal{T})}(\mathbf{x}_\beta)\right] \quad (13)$$

where we introduced Lagrange multipliers $Q(\mathbf{x})$ for the constraints (12). Differentiation with respect to $\phi_\beta^{(\mathcal{T})}(\mathbf{x}_\beta)$, using

$$\frac{\partial\log Z(\phi^{(\mathcal{T})})}{\partial\phi_\beta^{(\mathcal{T})}(\mathbf{x}_\beta)} = P^{(\mathcal{T})}(\mathbf{x}_\beta) \quad (14)$$

yields for $\beta \in C(\mathcal{T})$,

$$P^{(\mathcal{T})}(\mathbf{x}_\beta) = \sum_{\mathbf{x}\setminus\mathbf{x}_\beta}Q(\mathbf{x}) \equiv Q(\mathbf{x}_\beta) \quad (15)$$

Apparently, $Q(\mathbf{x})$ is a 'pseudo-probability' with 'pseudo-marginals' on the clusters of the junction trees that are equal to the cluster marginals of these junction trees. This implies that in the optimal $\{\phi\}$ all the cluster probabilities of the different junction trees are consistent on their overlaps.

$$P^{(\mathcal{T})}(\mathbf{x}_{\beta\cap\beta'}) = P^{(\mathcal{T}')}(\mathbf{x}_{\beta\cap\beta'}) = Q(\mathbf{x}_{\beta\cap\beta'}) \quad (16)$$

for any pair of clusters $\beta \in C(\mathcal{T})$ and $\beta' \in C(\mathcal{T}')$. We reformulate the $\log Z(\phi^{(\mathcal{T})})$ as $\min F(\{P_\gamma^{(\mathcal{T})}\})$. Using the fact that the optimal $\{P_\gamma^{(\mathcal{T})}\}$ are equal to the pseudo-marginals $Q(\mathbf{x}_\gamma)$, and using (12) we can add up of all the free energies

$$\sum_\mathcal{T}\mu(\mathcal{T})F^{(\mathcal{T})}(\{Q(\mathbf{x}_\gamma)\}) = \left[-\sum_\alpha\psi_\alpha(\mathbf{x}_\alpha)Q(\mathbf{x}_\alpha) + \sum_\mathcal{T}\mu(\mathcal{T})\sum_{\gamma\in\Gamma(\mathcal{T})}k_\gamma Q(\mathbf{x}_\gamma)\log Q(\mathbf{x}_\gamma)\right] \quad (17)$$

Introducing the counting numbers

$$c_\gamma = \sum_\mathcal{T}\mu(\mathcal{T})\sum_{\gamma'\in\Gamma(\mathcal{T})}k_{\gamma'}\delta_{\gamma\gamma'} \quad (18)$$

and writing $Q(X_\gamma) = Q_\gamma(X_\gamma)$ as independent pseudo-marginals we obtain the convexified Kikuchi free en-

Algorithm 1 Message-free RGBP.

initialize $Q_\alpha(\mathbf{x}_\alpha)$ and $Q_\delta(\mathbf{x}_\delta)$ as in (21)
repeat
  for all inner clusters δ **do**
    update $Q_\delta(x_\delta) \leftarrow Q_\delta^{\text{new}}(x_\delta)$ as in (23)
    for all outer clusters $\alpha \supset \delta$ **do**
      update $Q_\alpha(\mathbf{x}_\alpha) \leftarrow Q_\alpha^{\text{new}}(\mathbf{x}_\alpha)$ as in (24)
    end for
  end for
until convergence
return $Q_\alpha(\mathbf{x}_\alpha)$ and $Q_\delta(x_\delta)$

ergy

$$F_{\text{approx}} = -\sum_\alpha \sum_{\mathbf{x}_\alpha} \psi_\alpha(\mathbf{x}_\alpha) Q_\alpha(\mathbf{x}_\alpha) + \sum_\gamma \sum_{\mathbf{x}_\gamma} c_\gamma Q_\gamma(\mathbf{x}_\gamma) \log Q_\gamma(\mathbf{x}_\gamma) \quad (19)$$

which should be minimized under the constraint that the Q_γ's are consistent on their overlaps. The convexity follows by construction from the fact that F_{approx} is a convex combination of exact free energies which are convex.

3.2 RELATION WITH THE KIKUCHI FREE ENERGY

The Kikuchi free energy (Yedidia et al., 2001; Heskes et al., 2003) has the same functional form as in (19). The difference is in the counting numbers. In the Kikuchi free energy, the starting point is a set of outer clusters, typically coinciding with the clusters α of the original models (possibly after merging). The inner clusters δ are formed by taking all intersections of the outer clusters. The counting numbers follow from the recursive Moebius formula $c_\delta = 1 - \sum_{\gamma \supset \delta} c_\gamma$, with $c_\alpha = 1$ for all outer clusters. The standard Kikuchi free energy need not to be convex.

3.3 REWEIGHTED GENERALIZED BELIEF PROPAGATION

Due to the similarity between the Kikuchi free energy and its convexified version, the GBP algorithm for minimizing the Kikuchi free energy is easily generalized to reweighted generalized belief propagation (RGBP) for the convexified Kikuchi free energy. Here we describe a message-free form of RGBP. It is based on the message-free GBP presented in (Heskes and Zoeter, 2003)). The only difference is that we now allow $c_\alpha \neq 1$ for the outer clusters.

We divide the clusters $\gamma \in \cup_{\mathcal{T}} \Gamma(\mathcal{T})$ into 'outer clusters' coinciding with clusters α in the original model, and 'inner clusters' δ which are subclusters of α. The approximate free energy (19) can (loosely) be interpreted as the free-energy of a "junction tree-like pseudo-probability distribution",

$$\tilde{Q}(\mathbf{x}) = \frac{\prod_\alpha Q_\alpha(\mathbf{x}_\alpha)^{c_\alpha}}{\prod_\delta Q_\delta(x_\delta)^{-c_\delta}} \quad (20)$$

We start by initializing the inner and outer cluster marginals

$$Q_\alpha(\mathbf{x}_\alpha) \propto \exp(\frac{\psi_\alpha(\mathbf{x}_\alpha)}{c_\alpha}) \quad \text{and} \quad Q_\delta(\mathbf{x}_\delta) \propto 1\,. \quad (21)$$

so that $tQ(\mathbf{x})$ is proportional to the target distribution,

$$\tilde{Q}(\mathbf{x}) \propto \exp(\sum_\alpha \psi(\mathbf{x}_\alpha)) \quad (22)$$

Next we repeatedly update the inner and outer cluster pseudo-marginals that re-arrange information in the cluster marginals to make them mutually consistent while keeping (22) satisfied. The update rule for the inner clusters δ is

$$Q_\delta^{\text{new}}(\mathbf{x}_\delta) \propto Q_\delta(\mathbf{x}_\delta)^{\frac{c_\delta}{m_\delta + c_\delta}} \bar{Q}_\delta(\mathbf{x}_\delta)^{\frac{m_\delta}{m_\delta + c_\delta}} \quad (23)$$

with

$$m_\delta = \sum_{\alpha \supset \delta} c_\alpha \quad \text{and} \quad \bar{Q}_\delta(\mathbf{x}_\delta) \propto \left[\prod_{\alpha \supset \delta} Q_\alpha(\mathbf{x}_\delta)^{c_\alpha} \right]^{\frac{1}{m_\delta}}$$

The update rule for the outer clusters α is

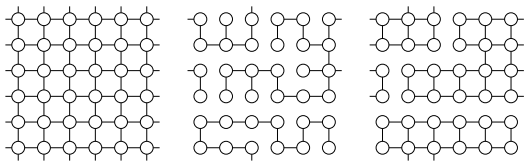
$$Q_\alpha^{\text{new}}(\mathbf{x}_\alpha) \propto Q_\alpha(\mathbf{x}_\alpha) \frac{Q_\delta^{\text{new}}(\mathbf{x}_\delta)}{Q_\alpha(\mathbf{x}_\delta)}\,. \quad (24)$$

The final RGBP algorithm is summarized in Algorithm 1.

In analogy with (Heskes and Zoeter, 2003) it can be shown that fixed points of Algorithm 1 is an extremum (and hence a minimum) of the convexified Kikuchi free energy. In our simulations, the algorithm converged without damping. To our knowledge, however, this is not guaranteed. If needed, a damping term can be introduced similar to the one in (Heskes and Zoeter, 2003).

4 COUNTING NUMBERS IN SOME REGULAR GRAPHS

For a graph with a random structure, the construction of the Kikuchi free energy is straightforward, given the choice of the clusters. The reason is that the counting numbers follow straightforwardly from the Moebius formula. In the convexified case, the computation of the counting numbers is generally much more



(a) (b) (c)

Figure 1: Left: full grid $T_{6\times 6}$ with periodic boundary conditions. Middle: spanning tree. Right: covering junction tree.

difficult (Wainwright et al., 2002). Since we are interested in the performance of nested convexified cluster approximations, without having a general algorithm for computing counting numbers, we restrict ourself to approximations of models with regular graphs, namely an Ising grid with periodic boundary conditions and a fully connected Ising model.

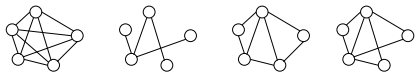
In (Wainwright et al., 2002), a convex combination of all spanning trees in the graph is taken, leading to a convexified Bethe free energy $F_{\rm cB}$. In this approximation the outer clusters are the pairs of connected nodes in the Ising model. In (Wainwright et al., 2002), the distribution $\mu(\mathcal{T})$ is optimized as well. Here we fix $\mu(\mathcal{T})$ to be uniform. Under this simplifying condition, and making use of the symmetries in the models that we consider, we can construct convexified Kikuchi free energies $F_{\rm cK}$ that will provide strictly tighter bounds

$$\min F \geq \min F_{\rm cK} \geq \min F_{\rm cB}. \qquad (25)$$

4.1 ISING GRID

The first Ising model that we consider is a grid of $2n \times 2n$ nodes with periodic boundary conditions (torus). Each node is connected to four neighbors. We denote this graph as $T_{2n\times 2n}$ (see Fig. 1(a)).

We consider pair approximations and 2×2 cluster approximations. To write down $F_{\rm cK}$, we have to compute the counting numbers c_γ. For this we draw a random junction tree with clusters and subclusters $\Gamma(\mathcal{T})$. For each type of cluster, namely singleton (type $\gamma(1)$), pair (type $\gamma(2)$), and 2×2 cluster (type $\gamma(4)$), we compute the sum of the counting numbers $k(i) = \sum_{\gamma \text{ is type } \gamma(i)} k_\gamma$ and divide the result by the total number $n(i)$ of clusters of type $\gamma(i)$ in the original graph. In the $T_{2n\times 2n}$ graph, these are $n(1) = 4n^2$, $n(2) = 8n^2$, and $n(4) = 4n^2$. The resulting counting number for clusters of type i is $c(i) = k(i)/n(i)$, which can be substituted in (19) for the c_γ's with γ of type i.



(a) (b) (c) (d)

Figure 2: Left: fully connected K_5. Middle: spanning tree. Right: two covering junction trees.

If we consider a spanning tree (see Fig. 1(b)), we find $k(2) = 4n^2 - 1$ and $k(1) = -(4n^2 - 2)$. As a result we find for $F_{\rm cB}$ the counting numbers

$$c(2) = \frac{4n^2-1}{8n^2} \quad \text{and} \quad c(1) = -\frac{2n^2-1}{2n^2} \qquad (26)$$

For our cluster approach we construct a set of junction trees as follows: take n non-overlapping horizontal strips of $2n-1$ type-4 clusters and connect these vertically by $n-1$ additional type-4 clusters (see Fig. 1(c)). Shifting and rotating this procedure leads to a homogeneous set. We find $k(4) = 2n^2-1$, $k(2) = -(2n^2-2)$, and $k(1) = 0$. So, with this choice of junction trees the counting numbers for $F_{\rm cK}$ are

$$c(4) = \frac{2n^2-1}{4n^2} \quad \text{and} \quad c(2) = -\frac{2n^2-1}{2n^2} \qquad (27)$$

If we now go back to the spanning trees and restrict the spanning trees to those that are contained in the junction trees as constructed above (which is the case in Fig. 1(b)), we find that the resulting counting numbers are the same as in (26). From this we conclude that the approximations are nested and (25) holds.

4.2 FULLY CONNECTED ISING MODEL

We denote the fully connected Ising model with n nodes as K_n (see Fig. 2(a)). We consider pair and triplet approximations. The counting numbers are needed for singletons (type $\gamma(1)$), pairs (type $\gamma(2)$), and triplets (type $\gamma(3)$). In the graph K_n, $n(1) = n$, $n(2) = n(n-1)/2$, and $n(3) = n(n-1)(n-2)/6$.

If we consider a spanning tree (see Fig. 2(b)), we find $k(2) = n - 1$ and $k(1) = -(n-2)$. The resulting counting numbers for $F_{\rm cB}$ are

$$c(2) = \frac{2}{n} \quad \text{and} \quad c(1) = -\frac{n-2}{n} \qquad (28)$$

For the triplet approximation we consider junction trees that have clusters of three nodes, and separators of two nodes (see Fig. 2(c,d)). For such type of junction trees we find $k(3) = n - 2$, $k(2) = -(n-3)$,

and $k(1) = 0$. So we find for F_{cK} the counting numbers

$$c(3) = \frac{6}{n(n-1)} \quad \text{and} \quad c(2) = -\frac{2(n-3)}{n(n-1)} \tag{29}$$

Each spanning tree is contained in a junction tree. Therefore the approximations are again nested and (25) holds.

5 SIMULATIONS

We apply (R)(G)BP to the Ising models described in the previous section. The prefix "G" (i.e, "generalized"), implies the use of outer-clusters larger than two, namely the 2×2 for the grids and the triplets for the fully connected models. The prefix "R" (i.e., "reweighted"), implies the use of counting numbers c_γ from the convexified free energy as described in the previous section rather than the standard ones obtained by the Moebius relation. For R(G)BP we used the RGBP algorithm as described earlier in this paper. For (G)BP we used the double-loop algorithm described in (Heskes et al., 2003).

5.1 EXPERIMENTAL SET-UP

We considered Ising grids $T_{6\times6}$ and $T_{8\times8}$ as well as the fully connected models K_9 and K_{12}. The variables in the models are binary $x_i = \pm 1$. We choose $\psi(\mathbf{x}_{ij}) = w_{ij}x_ix_j + \theta_i/n_ix_i + \theta_j/n_jx_j)$ with n_i the number of neighbors of node i. The external fields are generated according $\theta_i \sim \mathcal{N}(0, 0.01)$ (in both type of graphs) and couplings according to $J_{ij} \sim \mathcal{N}(0, \frac{\beta}{2})$ for the torus and $J_{ij} \sim \mathcal{N}(0, \frac{\beta}{\sqrt{N}})$, with N the number of nodes in the graph, for the fully connected model. We consider eight scalings $\beta = [0.2, 0.5, 0.75, 1, 1.5, 2, 5, 10]$. With each scaling, we generated 5 models. For each model realization we ran simulations with RBP, RGBP, BP and GBP. In all runs, we computed the exact minimal free energy $F_{\text{exact}} = -\log Z$, the exact edge probabilities $P(\mathbf{x}_{ij})$, the approximating free energy F_{approx} according to (19), and the approximating pseudo-marginals on the edges $Q(\mathbf{x}_{ij})$.

5.2 RESULTS

In figure 3 we plotted for the four models the maximum absolute deviation (MAD) of edge probabilities

$$\text{MAD} = \max_{(i,j)\in E} \max_{x_{ij}} |Q(\mathbf{x}_{ij}) - P(\mathbf{x}_{ij})|, \tag{30}$$

the relative error in free energy

$$\epsilon = \frac{F_{\text{approx}} - F_{\text{exact}}}{F_{\text{exact}}}, \tag{31}$$

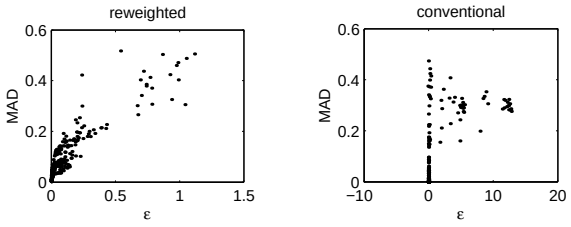


Figure 4: Scatter plot of performance in MAD as a function of relative error ϵ for reweighted approximations (RBP and RGBP, (left)) and conventional approximations (BP and GBP, (right)).

and the absolute values of these relative errors $|\epsilon|$. Plotted are the means and standard deviations of these quantities as function of interaction strength β.

On the grids $T_{6\times6}$ and $T_{8\times8}$, in both the conventional approximations (BP, GBP) as in the reweighted approximations (RBP,RGBP), increasing clustersize consistently improves the result: GBP outperforms BP, and RGBP outperforms RBP. In addition, we see that the conventional approximations (BP, GBP) outperform their reweighted counterparts (RBP, RGBP) both in the MAD and ϵ.

With the fully connected models K_9 and K_{12}, the results show a completely different picture for the conventional approximations: GBP performs only well for small β. If β is of order one, or larger, the GBP approximation collapses and BP outperforms GBP. (The large error bars in GBP for $\beta \approx 1$ are due to the fact that for some model realizations GBP performed very well, and for others very bad). On the other hand, adding complexity in the reweighted approximation always improves the result: RGBP is always better than RBP. The improvement is remarkably constant, almost independent of β, and independent of whether the approximation is good or bad.

To investigate the relation between ϵ and MAD of the two different classes of approximations, we pooled all the simulation results (i.e. of all the runs for the four models with all the settings of β) into two groups: one for the reweighted approximations (RBP and RGBP) and one for the conventional approximations (BP and GBP). In figure 4 we made scatter plots of each pool by plotting the MAD versus ϵ for each simulation run. In these plots we see that the relation between ϵ and MAD is much stronger in the the reweighted approximations than in the conventional approximations.

Furthermore, we investigated the effect of increasing the cluster size in each of the approximation classes. For each model realization, we compared the errors ϵ and MAD for the pair approximations BP and RBP

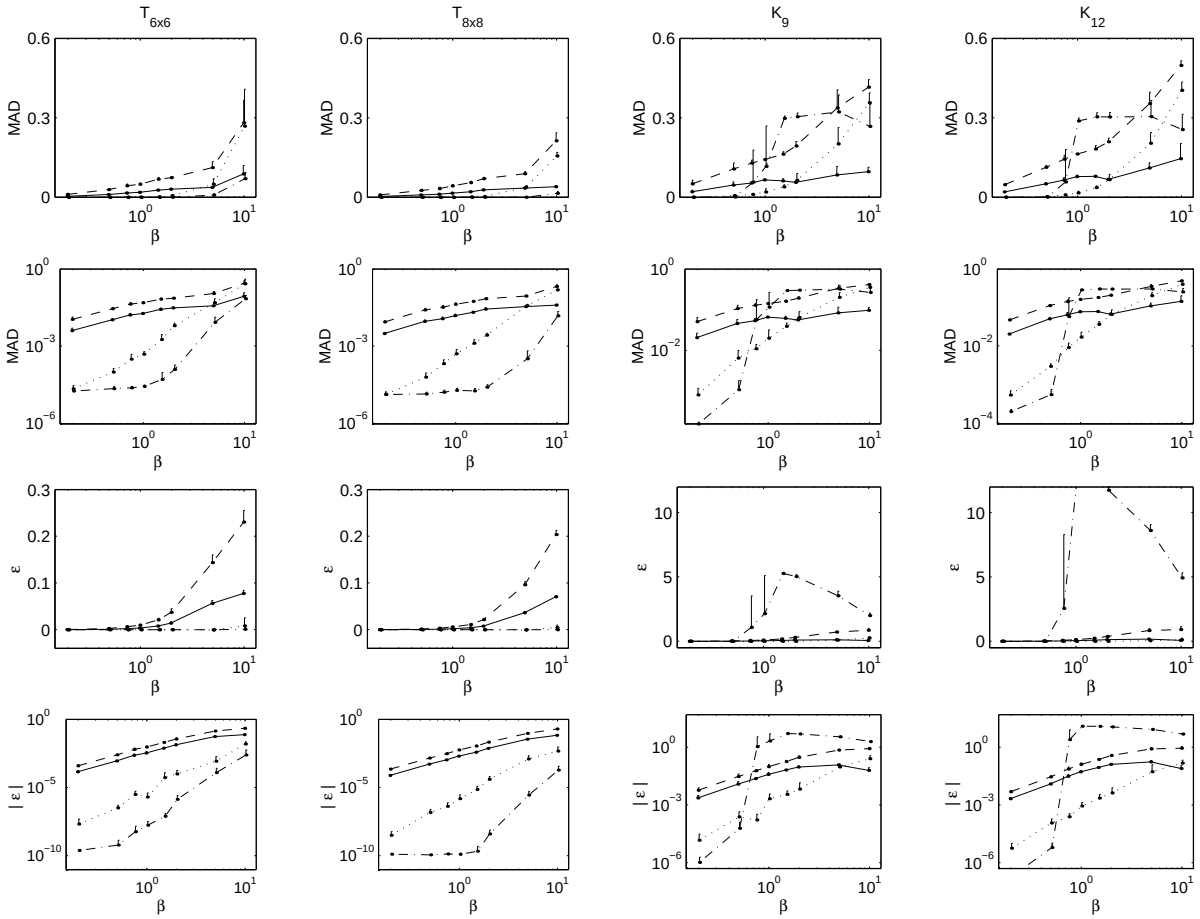


Figure 3: Columns from left to right: (a) Results for $T_{6\times6}$, (2 D grid with $6 \times 6 = 36$ nodes, periodic boundary conditions); (b) Results for $T_{8\times8}$. (c) Results for K_9 (fully connected model with 9 nodes). (d) Results for K_{12}. First row: Maximal absolute deviation (MAD) for edge probabilities $p(x_{ij})$ Second row:MAD, in log scale. Third row: Relative error in free energy $\epsilon = (F_{\text{approx}} - F_{\text{exact}})/F_{\text{exact}}$ in linear scale. Fourth row: absolute values of relative errors $|\epsilon|$ in log scale. Results are plotted for RGBP (full line), RBP (dashed), GBP (dash-dotted) and BP (dotted). Only upper parts of errorbars are drawn for visual clarity. (Large errorbars in K_9 and K_{12} are in GBP results).

with the cluster approximation in the same approximating class, GBP and RGBP respectively. The scatter plots in figure 5 clearly show that in the reweighted approximation class the performance with larger clusters improves in all model realizations. For the approximation in F, this improvement is theoretically guaranteed. The improvement in both ϵ and MAD is surprisingly constant. In the conventional approximation class, the improvement clearly depends on the regime. In the easy regime (small ϵ, small MAD), larger clusters improves the results. In the hard regime (large ϵ, large MAD), increasing cluster size may actually do harm. Furthermore, there are exceptions: sometimes BP improves upon GBP even in the easy regime.

6 DISCUSSION

Finding accurate approximations of graphical models such as Bayesian networks is crucial if their application to large scale problems is to be realized. Generalized belief propagation (GBP) is nowadays considered as one of the most powerful approximation methods. The method is flexible in the sense that there is a tradeoff in computational complexity and cluster-size. Unfortunately, increasing the cluster size does not guarantee to improve accuracy, and sometimes even deteriorate results. For this reason we are interested in alternative approximate methods that *do* provide a guarantee of improvement (at least in the free energy). Besides the convexified free energy approach, which is studied in this paper, another method that provide this guar-

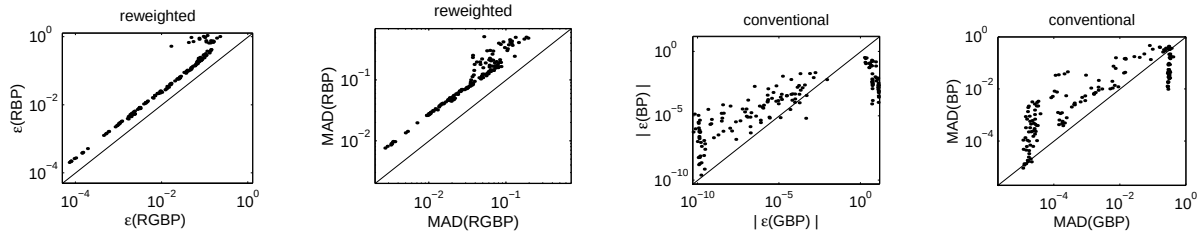


Figure 5: Performances (ϵ and MAD) of pair-approximation ((R)BP) versus performance of cluster-approximations (R)GBP.

antee is the structural mean field (SMF) approximation (Wiegerinck, 2000). The convexified free energy approach, however, has several appealing advantages. Unlike SMF theory, all the edges in the target distribution need (by construction) to be covered. This not only circumvents the problem in SMF of which edges to keep and which to delete, but it also suggests more powerful approximation. The additional fact that no results about its performance with cluster size larger than two have been published (as far as we know) motivated us to further investigate this method.

The experimental results with RGBP (reweighted generalized belief propagation -the counterpart of RBP for the convexified approach) suggest that in 'easy' problems where (G)BP performs well, the reweighted approximations do not provide a competing alternative. However, in 'hard' problems, it might be worthwhile to consider RGBP with larger clusters as an alternative. The method seems to be more robust in such problems.

There is, however, an important open problem, not addressed in this paper, but mentioned earlier in (Wainwright et al., 2002), which is: how to find – or even better: optimize the counting numbers in RGBP. In this paper, we computed them (suboptimally - since μ was taken constant) by hand, which was possible thanks to the symmetries in the model. An automatic procedure, however, is crucial if the RGBP is to be applied in real world problems with graphical models of arbitrary structure

Acknowledgments

This research is supported by the Dutch Technology Foundation STW. Thanks to Kees Albers for sharing his double-loop software.

References

Heskes, T., Albers, K., and Kappen, H. J. (2003). Approximate inference and constraint optimisation. In *UAI 19*, pages 313–320.

Heskes, T. and Zoeter, O. (2003). Generalized belief propagation for approximate inference in hybrid Bayesian networks. In *AISTATS 9*.

Jensen, F. (1996). *An introduction to Bayesian networks.* UCL Press.

Kappen, H. J. and Wiegerinck, W. (2002). Novel iteration schemes for the cluster variation method. In *NIPS 14*, pages 415–420.

Lauritzen, S. and Spiegelhalter, D. (1988). Local computations with probabilties on graphical structures and their application to expert systems (with discussion). *J. Royal Statistical Society Series B*, 50:157–224.

Murphy, K. P., Weiss, Y., and Jordan, M. I. (1999). Loopy belief propagation for approximate inference: An empirical study. In *UAI 15*, pages 467–475.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, Inc.

Teh, Y. and Welling, M. (2002). The unified propagation and scaling algorithm. In *NIPS 14*, pages 953–960.

Wainwright, M. J., Jaakkola, T., and Willsky, A. S. (2002). A new class of upper bounds on the log partition function. In *UAI 18*, pages 536–543.

Wainwright, M. J., Jaakkola, T., and Willsky, A. S. (2003). Tree-reweighted belief propagation and approximate ML estimation by pseudo-moment matching. In *AISTATS 9*.

Wainwright, M. J. and Jordan, M. I. (2003). Graphical models, exponential families, and variational inference. Technical Report 649, UC Berkeley, Dept. of Statistics.

Wiegerinck, W. (2000). Variational approximations between mean field theory and the junction tree algorithm. In *UAI 16*, pages 626–633.

Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2001). Generalized belief propagation. In *NIPS 13*, pages 689–695.

Yuille, A. L. (2002). CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation. *Neural computation*, 14:1691–1722.

Recursive Autonomy Identification for Bayesian Network Structure Learning

Raanan Yehezkel and Boaz Lerner
Pattern Analysis and Machine Learning Lab
Electrical and Computer Engineering
Ben-Gurion University, Israel
{*raanany, boaz@ee.bgu.ac.il*}

Abstract

We propose a constraint-based algorithm for Bayesian network structure learning called recursive autonomy identification (RAI). The RAI algorithm learns the structure by recursive application of conditional independence (CI) tests of increasing orders, edge direction and structure decomposition into autonomous sub-structures. In comparison to other constraint-based algorithms d-separating structures and then directing the resulted undirected graph, the RAI algorithm combines the two processes from the outset and along the procedure. Learning using the RAI algorithm renders smaller condition sets thus requires a smaller number of high order CI tests. This reduces complexity and run-time as well as increases accuracy since diminishing the curse-of-dimensionality. When evaluated on synthetic and "real-world" databases as well as the ALARM network, the RAI algorithm shows better structural correctness, run-time reduction along with accuracy improvement compared to popular constraint-based structure learning algorithms. Accuracy improvement is also demonstrated when compared to a common search-and-score structure learning algorithm.

1 INTRODUCTION

Most algorithms for Bayesian network (BN) structure learning are either search-and-score based [Heckerman, 1995; Friedman et al., 1997] in which the structure achieving the highest score given the data is pursued or constraint-based in which the structure is learned from constraints derived from statistical tests of independence between variables combined with causality inference rules [Pearl, 2000; Spirtes et al., 2000]. The main problem of constraint-based algorithms is their inefficiency and inaccuracy (due to the curse-of-dimensionality) in performing conditional independence (CI) tests for large condition sets. Most constraint-based algorithms, such as Inductive Causation (IC) [Pearl, 2000], PC [Spirtes et al., 2000] and Three Phase Dependency Analysis (TPDA), [Cheng et al., 1997], construct a directed acyclic graph (DAG) in two consecutive stages. First is learning associations between variables for constructing an undirected structure. This requires an exponentially growing number of CI tests with the number of nodes, which can be reduced to polynomial by fixing the number of parents (PC algorithm) or using the values computed in the CI test and some strong assumptions (TPDA algorithm). These assumptions however may not be valid in all situations. Another flaw of the TPDA algorithm is ignoring the curse-of-dimensionality in CI tests by not limiting the size of the condition set. The second stage in most constraint-based algorithms is causality inference performed in two consecutive steps: finding and directing V-structures and inductively directing additional edges [Pearl, 2000]. Causality inference, and especially the induction step, is unstable, i.e., small errors in the input to the stage yield large errors at its output [Spirtes et al., 2000]. Thus, the algorithms increase stability by separating the two stages trying in the first stage to minimize erroneous decisions about d-separation caused by invalid threshold selection or poor estimation due to the curse-of-dimensionality.

We propose a constraint-based algorithm that recursively tests conditional independencies with condition sets of increasing orders, directs edges for each order and identifies autonomous sub-structures complying with the Markov property (i.e., the sub-structure includes all node parents). By considering directed rather than undirected

edges, the RAI avoids unnecessary CI tests and performs tests using smaller condition sets. Repeated for autonomies decomposed recursively from the graph both mechanisms reduce computational and time complexities, database queries and errors of subsequent iterations. Using smaller condition sets, the RAI algorithm also improves accuracy since diminishing the curse-of-dimensionality. After providing some preliminaries in Section 2 we introduce the RAI algorithm in Section 3 and present its experimental evaluation in Section 4 before concluding the paper in Section 5.

2 PRELIMINARIES

A BN B(G,Θ) consists of a structure (graph) G and a set of probabilities Θ quantifying the graph. G(**V**,**E**) consists of **V**, a set of nodes representing domain variables, and **E** a set of edges connecting the nodes. $\mathbf{Pa_p}$(X,G), **Adj**(X,G) and **Ch**(X,G) are respectively the sets of potential parents, adjacent nodes and children of node X in a partially directed graph G, $\mathbf{Pa_p}(X,G)=\mathbf{Adj}(X,G)\backslash\mathbf{Ch}(X,G)$. Similarly, **Pa**(X,G) and **Desc**(X,G) are the sets of parents and descendants of X in G. We indicate that X and Y are independent given a set of nodes **S** using $X \perp\!\!\!\perp Y|\mathbf{S}$ and make use of the notion of d-separation [Pearl, 2000]. We also define d-separation resolution evaluating d-separation for different values of the maximal number of nodes in the condition set, an exogenous cause to a graph and an autonomous sub-structure.

Definition 1: The d-separation resolution between any pair of non-adjacent nodes is the size of the smallest condition set that d-separates the two nodes.

Definition 2: The d-separation resolution of a graph is the highest d-separation resolution in the graph.

Definition 3: Y is an exogenous cause to G(**V**,**E**) if $Y \notin \mathbf{V}$ and $\forall X \in \mathbf{V}$, $Y \in \mathbf{Pa}(X)$ or $Y \notin \mathbf{Adj}(X)$ [Pearl, 2000].

Definition 4: A sub-structure $G^A(\mathbf{V^A},\mathbf{E^A})$ in G(**V**,**E**) s.t $\mathbf{V^A} \subset \mathbf{V}$, $\mathbf{E^A} \subset \mathbf{E}$ is autonomous given a set of exogenous nodes $\mathbf{V_{ex}}$ to G^A if $\forall X \in \mathbf{V^A}$, $\mathbf{Pa}(X,G) \subset \{\mathbf{V^A} \cup \mathbf{V_{ex}}\}$. If $\mathbf{V_{ex}}$ is empty, we say the sub-structure is autonomous.

We define sub-structure autonomy in the sense that the sub-structure holds the Markov property for its nodes. Given a structure G, any two non-adjacent nodes in an autonomous sub-structure G^A are d-separated given nodes either included in the sub-structure or exogenous causes to it. This notion is elaborated in Section 3.3.

3 RECURSIVE AUTONOMY IDENTIFICATION

Starting from a complete graph and proceeding from low to high graph d-separation resolution, the RAI algorithm uncovers the correct pattern (i.e., a family of structures Markov equivalent to the true underlying structure) by recursive (1) test of CI between nodes and removal of edges related to independencies (thinning), (2) edge direction according to inferred causality rules and (3) graph decomposition into autonomous sub-structures.

CI testing of order n between X and Y is performed by thresholding a criterion, such as the χ^2 goodness of fit [Spirtes et al., 2000] or conditional mutual information [Cheng et al., 1997]. The criterion measures dependence conditioned on a set of n nodes from the parents of X or Y determined by the Markov property [Pearl, 2000], e.g., if X is directed into Y only Y's parents are included in the set.

Directing edges is conducted according to causality rules [Pearl, 2000] by identifying intransitive triplets of nodes (V-structures), i.e., non-adjacent parents having a common child, directing the relevant edges, and applying additional rules to further direct edges until no more edges can be directed (the inductive step).

Decomposition into autonomous sub-structures reveals the structure hierarchy and allows performing a fewer CI tests conditioned on a large number of potential parents thereby reducing complexity. The RAI algorithm identifies ancestor and descendant sub-structures, the latter are autonomous given nodes of the former.

3.1 THE RAI ALGORITHM

Iteration of the RAI algorithm starts with knowledge produced in the previous iteration and the current d-separation resolution, n. Previous knowledge includes G_{start}, a structure having d-separation resolution of n-1 and $\mathbf{G_{ex}}$, a set of structures having each possible exogenous causes to G_{start}. In the first iteration, n = 0, $G_{start}(\mathbf{V},\mathbf{E})$ is a complete graph and $\mathbf{G_{ex}}=\varnothing$.

Given a structure G_{start} having d-separation resolution n-1, the RAI algorithm seeks independencies between adjacent nodes conditioned on sets of size n, resulting in a structure having d-separation resolution of n. After directing edges, the algorithm decomposes the structure into ancestor and descendent autonomous sub-structures in order to reduce complexity of successive stages. A descendant sub-structure is established by identifying the lowest topological order nodes (either a single node or a

Main function G_{out} = RAI(n,G_{start},$\mathbf{G_{ex}}$)

Exit condition

If all nodes in G_{start} have less than n-1 potential parents exit.

A. Thinning the link between $\mathbf{G_{ex}}$ and $\mathbf{G_{start}}$ and directing $\mathbf{G_{start}}$

1. For every node Y in G_{start} and its parent X in $\mathbf{G_{ex}}$, if $\exists \mathbf{S} \subset \{\mathbf{Pa_p}(Y,\mathbf{G_{ex}}) \backslash X \cup \mathbf{Pa_p}(Y,G_{start})\}$ and $|\mathbf{S}|$=n s.t $X \perp\!\!\!\perp Y|\mathbf{S}$, then remove the edge between X and Y.
2. Direct the edges using causality inference rules.

B. Thinning, directing and decomposing $\mathbf{G_{start}}$.

1. For every node Y and its potential parent X, both in G_{start}, if $\exists \mathbf{S} \subset \{\mathbf{Pa_p}(Y,\mathbf{G_{ex}}) \cup \mathbf{Pa_p}(Y,G_{start}) \backslash X\}$ and $|\mathbf{S}|$=n s.t $X \perp\!\!\!\perp Y|\mathbf{S}$, then remove the edge between X and Y.
2. Direct the edges using causality inference rules.
3. Group the nodes having the lowest topological order into a descendant sub-structure G_D.
4. Remove G_D from G_{start} temporarily, and define the resulting unconnected structures as ancestor sub-structures $G_{A1},\ldots, G_{Ak}$.

C. Ancestor sub-structure decomposition

for i = 1 to k, call RAI(n+1,G_{Ai},$\mathbf{G_{ex}}$)

D. Descendant sub-structure decomposition

1. Define $\mathbf{G_{D\_ex}}$={$G_{A1},\ldots,G_{Ak},\mathbf{G_{ex}}$} as the exogenous structure to G_D.
2. Call RAI(n+1,G_D,$\mathbf{G_{D\_ex}}$)

Figure 1: The RAI algorithm

several nodes having the same lowest order). This structure is autonomous given ancestor sub-structures composed of nodes of higher order. In order to consider a smaller number of parents for each node of the descendent sub-structure, the algorithm recursively learns ancestor sub-structures and only then their descendant sub-structure. Note that this latter structure may consist of a several non-connected sub-structures. Figures 1-3 show respectively the RAI algorithm, a manifesting example and the algorithm execution order for this example. Figure 2a shows the true underlying structure. Initially, G_{start} is the complete graph and $\mathbf{G_{ex}}$ is empty so stage A is skipped. At stage B1, any pair of nodes is CI tested given an empty condition set (marginal independence) yielding the removal of the edges between node 1 and nodes 3, 4

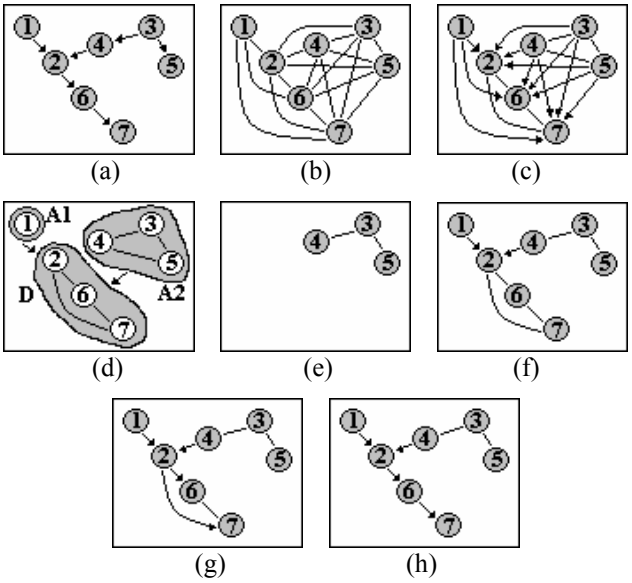


Figure 2: Learning an example structure. a) The true underlying structure and structures learned by the RAI algorithm in stages (see Figure 1) b) B1, c) B2, d) B4, e) C, f) D and A1, g) D and A2 and h) D and B1 (the resulting structure)

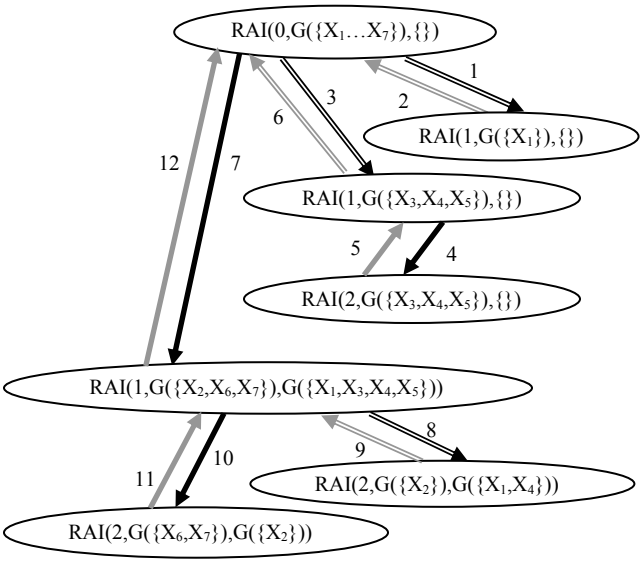


Figure 3: The execution order of the RAI algorithm for the structure of Figure 2. Recursive calls of stages C and D are marked with a double and single arrow, respectively.

and 5 (Figure 2b). The causal relations inferred at stage B2 are shown in Figure 2c. The nodes having the lowest topological order (2, 6, 7) are grouped into a descendant sub-structure G_D (stage B3) while the remaining nodes form two unconnected ancestor sub-structure, G_{A1} and G_{A2} (stage B4) (Figure 2d). At stage C the algorithm is called recursively for each of the ancestor sub-structures with n=1, $G_{start}=G_{Ai}$ and $\mathbf{G_{ex}}=\varnothing$. Since sub-structure G_{A1} contains a single node, the exit condition for the structure is satisfied. While calling $G_{start}=G_{A2}$, stage A is skipped and stage B1 identifies that $X_4 \perp\!\!\!\perp X_5|X_3$ thus removes edge X_4—X_5. No causal relations are identified so the nodes have equal topological order and they are grouped to from a descendant sub-structure. The recursive call for this sub-structure with n=2 is returned immediately since the exit condition is satisfied (Figure 2e). Moving to stage D, the RAI is called with n=1, $G_{start}=G_D$ and $\mathbf{G_{ex}}=\{G_{A1},G_{A2}\}$. Then, in stage A1 relations $(X_1 \perp\!\!\!\perp \{X_6,X_7\}|X_2)$, $(X_4 \perp\!\!\!\perp \{X_6,X_7\}|X_2)$ and $(\{X_3,X_5\} \perp\!\!\!\perp \{X_2,X_6,X_7\}|X_4)$ are identified and the corresponding edges are removed (Figure 2f). At stage A2 node X_2 is identified as a parent of X_6 and X_7 (Figure 2g). Stage B1 identifies the relation $(X_2 \perp\!\!\!\perp X_7|X_6)$ and stage B2 identifies X_6 as a parent of X_7 (Figure 2h). Further recursive calls are returned and the resulting partially directed structure represents a family of Markov equivalent structures of the true structure.

3.2 MINIMALITY, STABILITY & COMPLEXITY

Minimality A structure having a higher d-separation resolution entails a fewer dependencies and thus is simpler and preferred to a structure having a lower d-separation resolution [Pearl, 2000]. By increasing the resolution, the RAI algorithm moves from a complete structure having maximal dependency relations between variables to structures having less (or equal) dependencies than previous structures ending in a structure having no edges between conditionally independent nodes, i.e., a minimal structure.

Stability is measured by the number of errors in the output structure due to CI test errors, which are the only source of errors. CI test errors are the result of unnecessary large condition set leading to the curse-of-dimensionality or choosing an inaccurate condition set due to partial information (e.g., undirected edges). Although as a recursive algorithm the RAI might be unstable, errors are practically less likely to occur since the algorithm utilizes more information (e.g., edge direction and graph decomposition) from previous iterations to choose smaller, informative condition sets for performing the tests.

Complexity CI tests are the major contribution to complexity (run-time) [Cheng and Greiner, 1999]. In the worst case, the RAI algorithm will not direct any edges nor decompose the structure thus identify the entire structure as a descendant sub-structure calling stage D iteratively while skipping most other stages. Then, the execution of the algorithm will be similar to that of the PC algorithm and the complexity will be bounded by that of the PC algorithm. Given the maximal number of possible parents k and the number of nodes n, the number of CI tests is bounded by [Spirtes et al., 2000]

$$2\binom{n}{2}\cdot\sum_{i=0}^{k}\binom{n-1}{i}\leq\frac{n^2(n-1)^{k-1}}{(k-1)!}.$$

This worst case scenario rarely occurs in "real-world" applications requiring structures having colliders.

3.3 CORRECTNESS

Proposition: If the input data to the RAI algorithm is faithful to a DAG, G, having any d-separation resolution, then it yields the correct pattern, G_{out}.

Proof: (by induction, ignoring notions common to the RAI and PC algorithms which are proved in [Spirtes et al., 2000])

Base case: If the input data to the RAI algorithm is faithful to a DAG with d-separation resolution 0, then it yields the correct pattern G_{out}.

Since G_{start} is a complete graph, the algorithm tests in stage B marginal independence between pairs of nodes and then direct edges. Thus, the resulting structure contains only edges between marginally dependent nodes, therefore having d-separation resolution of 0. From the decomposition stages, B3 and B4, based on the topological order identified from the partially directed structure, it follows that every edge from a node X in an ancestor sub-structure to a node Z in the descendant sub-structure is directed, $X \rightarrow Z$. Also, there is no edge connecting one ancestor sub-structure to another ancestor sub-structure. Thus, every ancestor sub-structure contains all the potential parents of its nodes, i.e., it is autonomous.

Lemma 1: If the given data entails $X \perp\!\!\!\perp Y|\mathbf{S}$ and X,Y are members of an autonomous sub-structure $G^A(\mathbf{V^A},\mathbf{E^A})$, then $\exists\mathbf{S'}$ such that $\mathbf{S'}\subset\mathbf{V^A}$ and $X \perp\!\!\!\perp Y|\mathbf{S'}$.

Lemma 2: In a DAG, if X and Y are non-adjacent and X is not a descendant of Y then X and Y are d-separated given **Pa**(Y) (proved in [Spirtes et al., 2000]).

An autonomous sub-structure contains all potential parents (either sub-structure nodes or exogenous causes) of each of its nodes. Thus, from Lemma 2, if X and Y are independent given a set of nodes (i.e., d-separated in the true underlying graph), then they are d-separated given $\mathbf{Pa_P}(X)$ or $\mathbf{Pa_P}(Y)$ which are contained in the autonomous sub-structure. Thus, every ancestor sub-structure can be processed independently by recursive calls of the algorithm. The recursive call of the descendant sub-structure regards the ancestor sub-structure nodes as exogenous causes. The data does not entail any higher order conditional independencies and no more edges are removed.

Inductive case: Suppose that the RAI algorithm yields the correct pattern given data faithful to a DAG having d-separation resolution n. Then, given data faithful to a DAG having d-separation resolution n+1 the RAI algorithm yields the correct pattern.

After achieving d-separation resolution of n in an autonomous sub-structure, $G^{(n)}$, a recursive call with n+1 is made. The exit condition is not satisfied in case an edge exists in $G^{(n)}$ and does not exist in the true structure G_t. Suppose an edge $E_{XY}=(X\rightarrow Y)$ exists, such that $E_{XY}\in G^{(n)}$ and $E_{XY}\notin G_t$, then the smallest condition set required to identify the independency between the nodes is S_{XY}, such that $|S_{XY}| \geq n+1$. Thus, it follows from Lemma 2 that either $|\mathbf{Pa}(X)| \geq n+1$ or $|\mathbf{Pa}(Y)|\geq n+1$ and the exit condition is not satisfied. Every pair of connected nodes is tested for independence in stage B1 using condition sets of size n+1 and the corresponding edges are removed resulting in a sub-structure having d-separation resolution of n+1.

The correctness of edge directing is discussed in [Pearl, 2000; Spirtes et al., 2000].

4 EXPERIMENTS AND RESULTS

The RAI algorithm was experimentally compared to the PC and TPDA algorithms, two popular constraint-based structure learning algorithms reported frequently as having good performance [Ramsey et al., 2002]. For simplicity, no speeding-up heuristic techniques [Spirtes et al., 2000] were applied to either algorithm, and the RAI algorithm employed only V-structure identification deferring the inductive step after forming the structure.

The complexity and prediction accuracy of the RAI algorithm were compared to those of the PC and TPDA algorithms using a synthetic problem and fifteen "real-world" databases of the UCI Repository [Murphy and Aha, 1994]. Interested mainly in classification, the

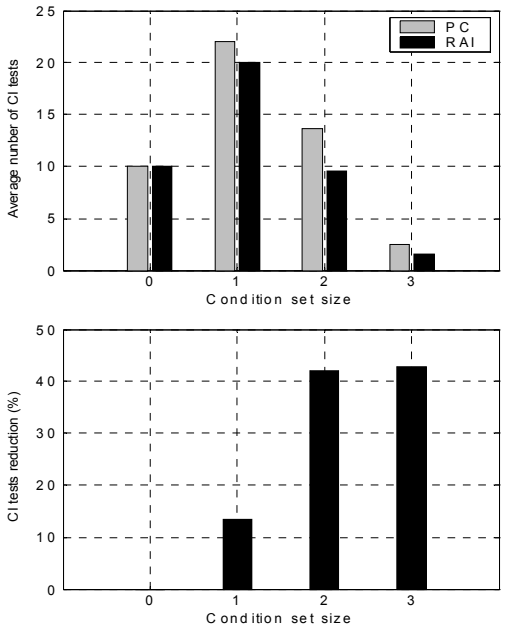


Figure 4: (a) The number of CI tests required by the RAI and PC algorithms for increasing orders averaged over all possible structures having five nodes. (b) CI test reduction by the RAI algorithm compared to the PC algorithm

prediction accuracy is preferred over the likelihood in evaluating performance, as the likelihood ignores the importance of the class variable [Friedman et al., 1997]. Structural correctness was evaluated in recovering the ALARM network in comparison to the TPDA and PC algorithms. BN implementation was aided by the Bayes net toolbox (BNT) [Murphy, 2001] and BNT structure learning package [Leray and Francois, 2004].

4.1 A SYNTHETIC PROBLEM

All 29,281 possible structures having five nodes were learned by the PC and RAI algorithms. Since the true structure is known, the actual CI relationships could be inputted to the algorithms. Figure 4a shows the complexity, evaluated using the averaged number of CI tests over all possible structures, of the algorithms for increasing orders (condition sets). Figure 4b illustrates the percentage of CI tests reduced by the RAI algorithm in comparison to the PC algorithm.

4.2 "REAL-WORLD" DATA

A several databases of the UCI Repository were employed in order to evaluate prediction accuracy. When needed, continuous variables were discretized using the

Table 1. The average number (percentage) of CI tests reduced by the RAI algorithm compared to the PC algorithm for different orders

Database	CI test order				
	0	1	2	3	4
shuttle (s)	0 (0)	1.4 (0.7)	95.8 (43.8)	117.6 (49.3)	83.6 (56.0)
car	0 (0)	16 (100)	11.2 (100)	3.2 (100)	
corral	0 (0)	22.4 (100)	26 (100)	3.6 (100)	
mofn 3,7,10	0 (0)	17 (100)	4 (100)		
tic-tac-toe	0 (0)	53.2 (27.1)	56.6 (48.6)	1.8 (51.4)	
led7	0 (0)	46.2 (45.7)	105 (100)	140 (100)	105 (100)
breast	0 (0)	107.2 (54.8)	35 (99.1)		
vote	0 (0)	24.2 (21.9)	17.2 (98.1)	6.4 (100)	1 (100)
flare C	0 (0)	16 (39.6)	3 (100)		
wine	0 (0)	25.8 (41.0)	44.2 (67.6)	40.6 (82.4)	19 (96.7)
cmc	0 (0)	10.2 (10.9)	8 (32.5)		
crx	0 (0)	8.8 (49.6)			
zoo	0 (0)	82 (27.8)	365.8 (29.6)	1033.4 (27.7)	1928.6 (25.6)
australian	0 (0)	3.8 (34.4)			
iris	0 (0)	2 (40)			

Table 2. Mean (std) prediction accuracy of the RAI algorithm in comparison to the PC algorithm and "other" classifiers reported in [Friedman et al., 1997] (F) and [Cheng and Greiner, 1999] (TPDA algorithm) (C), as well as the cut (%) of CI test run-time using the RAI algorithm in comparison to the PC algorithm

Database	run-time cut (%)	PC accuracy (%)	RAI accuracy (%)	other
shuttle (s)	**38.94**	98.40	**99.22**	99.17(F)
car	**91.10**	85.07 (1.83)	**92.94** (1.06)	86.11(C)
corral	**87.94**	84.53 (15.45)	**98.52** (3.31)	97.60(F)
mofn 3,7,10	**67.70**	81.45	**93.16**	85.94(F)
tic-tac-toe	**36.52**	74.74 (1.48)	**75.57** (1.93)	
led7	**91.74**	73.31 (1.80)	**73.59** (1.56)	
breast	**71.87**	95.46 (2.04)	96.49 (1.61)	**96.92**(F)
vote	**46.06**	95.64 (1.87)	**95.87** (1.71)	94.94(F) 95.17(C)
flare C	**20.38**	**84.30** (2.54)	**84.30** (2.54)	82.74(F) 82.27(C)
wine	**29.11**	85.44 (7.79)	**87.07** (5.88)	
cmc	**14.22**	50.92 (2.33)	**51.12** (3.16)	
crx	**25.25**	**86.38** (2.63)	**86.38** (2.63)	85.60(F)
zoo	**13.63**	88.95 (8.79)	88.95 (8.79)	
australian	**6.05**	85.51 (0.52)	85.51 (0.52)	**86.23**(F)
iris	**19.10**	**96.00** (4.35)	93.33 (2.36)	94.00(F)

MLC++ library [Kohavi et al., 1994]. Variable A14 of the "shuttle-small (s)" database was ignored by the discretization function of MLC++ and thus omitted from the experiments. "flare1" and "flare2" were merged to form the "flare C" database where the class node is the number of "C-class" flares. All databases were analyzed using a CV5 experiment except the large "shuttle" and "mofn 3-7-10" databases which were analyzed using a hold-out experiment. CI tests were carried out using the χ^2 test [Spirtes et al., 2000] with thresholds chosen for each algorithm and database in order to maximize the prediction accuracy on a validation set selected from the training set. If a several thresholds were suitable, the chosen threshold was that leading to the fewest CI tests. Parameter learning was performed using sequential Bayesian updating with Dirichlet priors of unit hyper-parameters [Heckerman, 1995].

Complexity was measured by the number of CI tests employed and the corresponding run-time. Table 1 shows the average number and percentage of CI tests reduced by the RAI algorithm compared to the PC algorithm for

different orders. A 100% cut in CI tests for a specific order means that the RAI does not need any CI tests for this order. Empty cells mean that no CI tests of this order are required. Both Table 1 and Table 2, depicting the cut in run-time due to the RAI algorithm, demonstrate that the RAI algorithm outperforms the PC algorithm in all cases.

Prediction Accuracies of the RAI and PC algorithms for the experimented databases are summarized in Table 2. On ten of the fifteen databases the RAI algorithm improves accuracy on the PC algorithm, on four keeps accuracy intact and on the remaining "iris" database deteriorates accuracy. Examination of the "iris" database reveals discrepancy between the results of CI tests of orders 0 and 1 violating the Markov property. Three nodes are found marginally dependent on each other whereas nodes of each pair of this triplet are found independent given the third node. The prediction accuracy is also compared in Table 2 to that of the TPDA algorithm [Cheng and Greiner, 1999] and a BN learned by a search-and-score method using the minimum description length criterion [Friedman et al., 1997].

4.3 LEARNING THE ALARM NETWORK

Recovering the correct structure was evaluated using the ALARM network [Beinlich et al., 1989], which is widely accepted as a benchmark for evaluating structure learning algorithms. The RAI algorithm was compared to the PC and TPDA (PowerConstructor [Cheng, 1998]) algorithms using ten randomly generated databases each containing 10,000 cases. Since the TPDA algorithm had used the conditional mutual information CI test, we employed this test also here. For comparison, we selected the TPDA threshold of 0.003 [Cheng et al., 1997] for testing also the RAI algorithm and a threshold of 0.002 for the PC algorithm providing better accuracy for this algorithm than using a threshold of 0.003.

Structural correctness for the algorithms was evaluated using two types of errors due to extra edges (EE; commission) and missing edges (ME; omission) (Table 3). The PC and RAI algorithms achieved the smallest errors of extra and missing edges, respectively. The total structural error (Table 4) accounting for both errors was evaluated using

$$\text{Error}_T = \sqrt{\text{EE}^2 + \text{ME}^2}\ .$$

The RAI algorithm yielded structures with the smallest total structural error of all algorithms which was validated using a t-test with 1% significance level. Others structural errors (e.g., edge reversal) were not recorded though we expect the RAI algorithm to dominate both algorithms due to its enhanced mechanism of directing edges.

Table 3. Extra edge (EE) and missing edge (ME) errors (%) when learning the ALARM network in 10 trials using the TPDA, PC and RAI algorithms

Trial	TPDA		PC		RAI	
	EE	ME	EE	ME	EE	ME
1	0.48	8.70	0.16	2.17	0.97	0
2	0.32	4.35	0	6.52	0.65	2.17
3	0.32	4.35	0	4.35	0.65	2.17
4	0.32	6.52	0.16	4.35	0.32	0
5	0.48	8.70	0	2.17	0.65	0
6	0.48	8.70	0.16	4.35	0.48	0
7	0.48	8.70	0.32	0	0.65	0
8	0.16	2.17	0.16	2.17	0.81	2.17
9	0.16	2.17	0.16	4.35	0.81	2.17
10	0.48	8.70	0.32	4.35	0.65	0
mean (std)	0.37 (0.13)	6.30 (2.80)	**0.15** (0.12)	3.48 (1.83)	0.66 (0.18)	**0.87** (1.12)

Table 4. The total structural error (%) in 10 trials of the ALARM network learned using the TPDA, PC and RAI algorithms

Trial	TPDA	PC	RAI
1	8.71	2.18	0.97
2	4.36	6.52	2.27
3	4.36	4.35	2.27
4	6.53	4.35	0.32
5	8.71	2.17	0.65
6	8.71	4.35	0.48
7	8.71	0.32	0.65
8	2.18	2.17	2.32
9	2.18	4.35	2.32
10	8.71	4.36	0.65
mean (std)	6.32 (2.80)	3.51 (1.77)	**1.29** (0.88)

Complexity The average reduction in CI tests achieved by the RAI algorithm compared to the PC algorithm for the ALARM network is presented in Figure 5. The RAI algorithm avoids completely the use of CI tests of order 4 and 5 and almost completely CI tests of order 3, and it reduces the use of CI tests of order 2 by more than 83%. However, there is almost no reduction in CI tests of order 1 which are most of the tests. The total CI test run-time

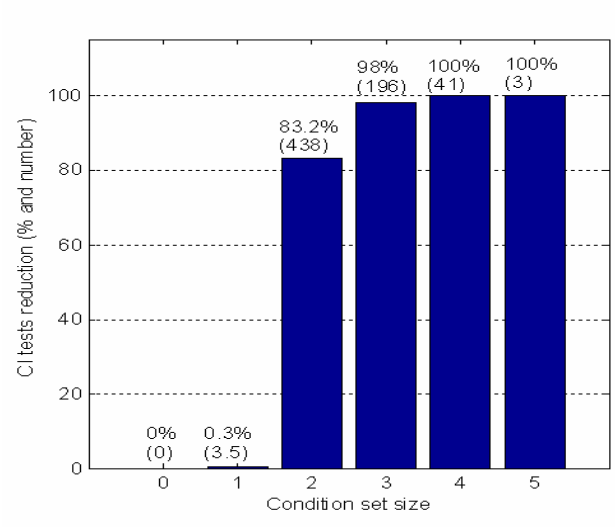


Figure 5: Average percentage (number) of CI tests reduced due to the RAI algorithm compared to the PC algorithm for increasing orders and the ALARM network

reduced by the RAI algorithm compared to the PC algorithm is 38%.

5 DISCUSSION

The performance of constraint-based algorithms of BN structure learning depends on the size of the condition set used for testing conditional independence. The larger the condition set is, the more CI tests (especially of high order) have to be performed and the less is their accuracy.

We propose the constraint-based RAI algorithm that learns BN structures recursively by performing 1) CI tests of increasing orders, along with 2) directing edges employing causality inference rules and 3) decomposing the structure into autonomous sub-structures. These mechanisms provide smaller condition sets enabling the performance of fewer CI tests of higher order thus reducing the algorithm run-time and increasing its accuracy. Other constraint-based algorithms directing edges after accomplishing the undirected graph using all orders, rather than continuously through learning, are expensive and more sensitive to errors accumulated along the procedure.

We demonstrate on a synthetic problem, fifteen real-world databases and the ALARM network that the RAI algorithm significantly reduces the number of CI tests required for structure learning and yields more accurate structures as well as higher prediction accuracy compared to other constraint-based algorithms.

Acknowledgement

This work was supported in part by the Paul Ivanier Center for Robotics and Production Management, Ben-Gurion University, Beer-Sheva, Israel.

References

Beinlich, I. A., Suermondt, H. J., Chavez, R. M. & Cooper, G. F. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. *Second European Conf. on Artificial Intelligence in Medicine*, pages 246-256, 1989.

Cheng, J. PowerConstructor system, 1998. http://www.cs.ualberta.ca/~jcheng/bnpc.htm.

Cheng, J., Bell, D. & Liu, W. Learning Bayesian networks from data: an efficient approach based on information theory. *Sixth ACM Int. Conf. on Information and Knowledge Management*, pages 325-331, 1997.

Cheng, J. & Greiner, R. Comparing Bayesian network classifiers, *Fifteenth Conf. on Uncertainty in Artificial Intelligence*, pages 101-107, 1999.

Friedman, N., Geiger, D. & Goldszmidt, M. Bayesian network classifiers. *Machine Learning,* 29:131-161, 1997.

Heckerman, D. A tutorial on learning with Bayesian networks. MS TR-95-06, March 1995.

Kohavi, R., John, G., Long, R., Manley D. & Pfleger, K. MLC++: A machine learning library in C++, *Sixth Int. Conf. on Tools with AI*, pages 740-743, 1994.

Leray, P. & Francois, O. BNT structure learning package: documentation and experiments. PSI TR, 2004.

Murphy, K. Bayes net toolbox for Matlab. Computing Science & Statistics, 33, 2001.

Murphy, P. M. & Aha, D. W. UCI Repository of machine learning databases, 1994. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Pearl, J. *Causality: Models, Reasoning, and Inference*. Cambridge. 2000.

Ramsey, J., Gazis, P., Roush, T., Spirtes, P. & Glymour, C. Automated remote sensing with near infrared reflectance spectra: Carbonate recognition. *Data Mining & Knowledge Discovery*, pages 277-293, 2002.

Spirtes, P., Glymour, C. & Scheines, R. *Causation, Prediction and Search*, 2nd edition, MIT Press, 2000.

Dirichlet Enhanced Latent Semantic Analysis

Kai Yu
Siemens Corporate Technology
D-81730 Munich, Germany
Kai.Yu@siemens.com

Shipeng Yu
Institute for Computer Science
University of Munich
D-80538 Munich, Germany
spyu@dbs.informatik.uni-muenchen.de

Volker Tresp
Siemens Corporate Technology
D-81730 Munich, Germany
Volker.Tresp@siemens.com

Abstract

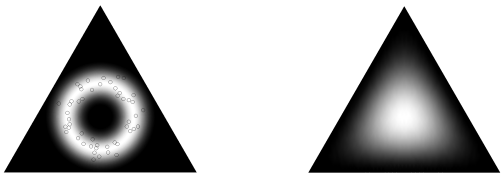
This paper describes nonparametric Bayesian treatments for analyzing records containing occurrences of items. The introduced model retains the strength of previous approaches that explore the latent factors of each record (e.g. topics of documents), and further uncovers the clustering structure of records, which reflects the statistical dependencies of the latent factors. The nonparametric model induced by a *Dirichlet process* (DP) flexibly adapts model complexity to reveal the clustering structure of the data. To avoid the problems of dealing with infinite dimensions, we further replace the DP prior by a simpler alternative, namely *Dirichlet-multinomial allocation* (DMA), which maintains the main modelling properties of the DP. Instead of relying on Markov chain Monte Carlo (MCMC) for inference, this paper applies efficient variational inference based on DMA. The proposed approach yields encouraging empirical results on both a toy problem and text data. The results show that the proposed algorithm uncovers not only the latent factors, but also the clustering structure.

1 Introduction

We consider the problem of modelling a large corpus of high-dimensional discrete records. Our assumption is that a record can be modelled by latent factors which account for the co-occurrence of items in a record. To ground the discussion, in the following we will identify records with documents, latent factors with (latent) topics and items with words. Probabilistic latent semantic indexing (PLSI) [7] was one of the first approaches that provided a probabilistic approach towards modelling text documents as being composed of latent topics. Latent Dirichlet allocation (LDA) [3] generalizes PLSI by treating the topic mixture parameters (i.e. a multinomial over topics) as variables drawn from a Dirichlet distribution. Its Bayesian treatment avoids overfitting and the model is generalizable to new data (the latter is problematic for PLSI). However, the parametric Dirichlet distribution can be a limitation in applications which exhibit a richer structure. As an illustration, consider Fig. 1 (a) that shows the empirical distribution of three topics. We see that the probability that all three topics are present in a document (corresponding to the center of the plot) is near zero. In contrast, a Dirichlet distribution fitted to the data (Fig. 1 (b)) would predict the highest probability density for exactly that case. The reason is the limiting expressiveness of a simple Dirichlet distribution.

This paper employs a more general nonparametric Bayesian approach to explore not only latent topics and their probabilities, but also complex dependencies between latent topics which might, for example, be expressed as a complex clustering structure. The key innovation is to replace the parametric Dirichlet prior distribution in LDA by a flexible nonparametric distribution $G(\cdot)$ that is a sample generated from a *Dirichlet process* (DP) or its finite approximation, *Dirichlet-multinomial allocation* (DMA). The Dirichlet distribution of LDA becomes the base distribution for the Dirichlet process. In this *Dirichlet enhanced* model, the posterior distribution of the topic mixture for a new document converges to a flexible mixture model in which both mixture weights and mixture parameters can be learned from the data. Thus the *a posteriori* distribution is able to represent the distribution of topics more truthfully. After convergence of the learning procedure, typically only a few components with non-negligible weights remain; thus the model is able to naturally output clusters of documents.

Nonparametric Bayesian modelling has attracted considerable attentions from the learning community



(a) (b)

Figure 1: Consider a 2-dimensional simplex representing 3 topics (recall that the probabilities have to sum to one): (a) We see the probability distribution of topics in documents which forms a ring-like distribution. Dark color indicates low density; (b) The 3-dimensional Dirichlet distribution that maximizes the likelihood of samples.

(e.g. [1, 13, 2, 15, 17, 16]). A potential problem with this class of models is that inference typically relies on MCMC approximations, which might be prohibitively slow in dealing with the large collection of documents in our setting. Instead, we tackle the problem by a less expensive variational mean-field inference based on the DMA model. The resultant updates turn out to be quite interpretable. Finally we observed very good empirical performance of the proposed algorithm in both toy data and textual document, especially in the latter case, where meaningful clusters are discovered.

This paper is organized as follows. The next section introduces Dirichlet enhanced latent semantic analysis. In Section 3 we present inference and learning algorithms based on a variational approximation. Section 4 presents experimental results using a toy data set and two document data sets. In Section 5 we present conclusions.

2 Dirichlet Enhanced Latent Semantic Analysis

Following the notation in [3], we consider a corpus $\mathcal{D}$ containing D documents. Each document d is a sequence of N_d words that is denoted by $\mathbf{w}_d = \{w_{d,1}, \ldots, w_{d,N_d}\}$, where $w_{d,n}$ is a variable for the n-th word in $\mathbf{w}_d$ and denotes the index of the corresponding word in a vocabulary V. Note that a same word may occur several times in the sequence $\mathbf{w}_d$.

2.1 The Proposed Model

We assume that each document is a mixture of k latent topics and words in each document are generated by repeatedly sampling topics and words using the distributions

$$w_{d,n}|z_{d,n};\beta \sim \text{Mult}(z_{d,n},\beta) \quad (1)$$
$$z_{d,n}|\theta_d \sim \text{Mult}(\theta_d). \quad (2)$$

$w_{d,n}$ is generated given its latent topic $z_{d,n}$, which takes value $\{1,\ldots,k\}$. β is a $k \times |V|$ multinomial parameter matrix, $\sum_j \beta_{i,j} = 1$, where $\beta_{z,w_{d,n}}$ specifies the probability of generating word $w_{d,n}$ given topic z. θ_d denotes the parameters of a multinomial distribution of document d over topics for $\mathbf{w}_d$, satisfying $\theta_{d,i} \geq 0, \sum_{i=1}^{k} \theta_{d,i} = 1$.

In the LDA model, θ_d is generated from a k-dimensional Dirichlet distribution $G_0(\theta) = \text{Dir}(\theta|\lambda)$ with parameter $\lambda \in \mathbb{R}^{k\times 1}$. In our Dirichlet enhanced model, we assume that θ_d is generated from distribution $G(\theta)$, which itself is a random sample generated from a *Dirichlet process* (DP) [5]

$$G|G_0,\alpha_0 \sim \text{DP}(G_0,\alpha_0), \quad (3)$$

where nonnegative scalar α_0 is the *precision parameter*, and $G_0(\theta)$ is the *base distribution*, which is identical to the Dirichlet distribution. It turns out that the distribution $G(\theta)$ sampled from a DP can be written as

$$G(\cdot) = \sum_{l=1}^{\infty} \pi_l \delta_{\theta_l^*}(\cdot) \quad (4)$$

where $\pi_l \geq 0$, $\sum_l^{\infty} \pi_l = 1$, $\delta_\theta(\cdot)$ are point mass distributions concentrated at θ, and θ_l^* are countably infinite variables i.i.d. sampled from G_0 [14]. The probability weights π_l are solely depending on α_0 via a *stick-breaking process*, which is defined in the next subsection. The generative model summarized by Fig. 2(a) is conditioned on $(k \times |V| + k + 1)$ parameters, i.e. β, λ and α_0.

Finally the likelihood of the collection $\mathcal{D}$ is given by

$$\mathcal{L}_{\text{DP}}(\mathcal{D}|\alpha_0,\lambda,\beta) = \int_G \left\{ p(G;\alpha_0,\lambda) \prod_{d=1}^{D} \int_{\theta_d} \left[p(\theta_d|G) \prod_{n=1}^{N_d} \sum_{z_{d,n}=1}^{k} p(w_{d,n}|z_{d,n};\beta) p(z_{d,n}|\theta_d) \right] d\theta_d \right\} dG. \quad (5)$$

In short, G is sampled once for the whole corpus $\mathcal{D}$, θ_d is sampled once for each document d, and topic $z_{d,n}$ sampled once for the n-th word $w_{d,n}$ in d.

2.2 Stick Breaking and Dirichlet Enhancing

The representation of a sample from the DP-prior in Eq. (4) is generated in the stick breaking process in which infinite number of pairs (π_l, θ_l^*) are generated.

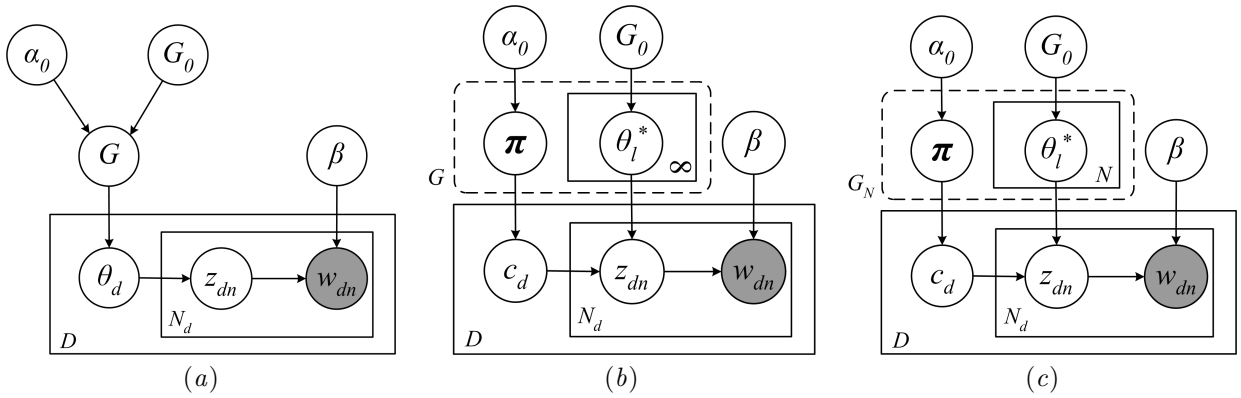


Figure 2: Plate models for latent semantic analysis. (a) Latent semantic analysis with DP prior; (b) An equivalent representation, where c_d is the indicator variable saying which cluster document d takes on out of the infinite clusters induced by DP; (c) Latent semantic analysis with a finite approximation of DP (see Sec. 2.3).

θ_l^* is sampled independently from G_0 and π_l is defined as

$$\pi_1 = B_1, \quad \pi_l = B_l \prod_{j=1}^{l-1}(1 - B_j),$$

where B_l are i.i.d. sampled from Beta distribution $\text{Beta}(1, \alpha_0)$. Thus, with a small α_0, the first "sticks" π_l will be large with little left for the remaining sticks. Conversely, if α_0 is large, the first sticks π_l and all subsequent sticks will be small and the π_l will be more evenly distributed. In conclusion, the base distribution determines the locations of the point masses and α_0 determines the distribution of probability weights. The distribution is nonzero at an infinite number of discrete points. If α_0 is selected to be small the amplitudes of only a small number of discrete points will be significant. Note, that both locations and weights are not fixed but take on new values each time a new sample of G is generated. Since $\mathbb{E}(G) = G_0$, initially, the prior corresponds to the prior used in LDA. With many documents in the training data set, locations θ_l^* which agree with the data will obtain a large weight. If a small α_0 is chosen, parameters will form clusters whereas if a large α_0, many representative parameters will result. Thus Dirichlet enhancement serves two purposes: it increases the flexibility in representing the posterior distribution of mixing weights and encourages a clustered solution leading to insights into the document corpus.

The DP prior offers two advantages against usual document clustering methods. First, there is no need to specify the number of clusters. The finally resulting clustering structure is constrained by the DP prior, but also adapted to the empirical observations. Second, the number of clusters is not fixed. Although the parameter α_0 is a control parameter to tune the tendency for forming clusters, the DP prior allows the creation of new clusters if the current model cannot explain upcoming data very well, which is particularly suitable for our setting where dictionary is fixed while documents can be growing.

By applying the stick breaking representation, our model obtains the equivalent representation in Fig. 2(b). An infinite number of θ_l^* are generated from the base distribution and the new indicator variable c_d indicates which θ_l^* is assigned to document d. If more than one document is assigned to the same θ_l^*, clustering occurs. $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_\infty\}$ is a vector of probability weights generated from the stick breaking process.

2.3 Dirichlet-Multinomial Allocation (DMA)

Since infinite number of pairs (π_l, θ_l^*) are generated in the stick breaking process, it is usually very difficult to deal with the unknown distribution G. For inference there exist Markov chain Monte Carlo (MCMC) methods like Gibbs samplers which directly sample θ_d using Pólya urn scheme and avoid the difficulty of sampling the infinite-dimensional G [4]; in practice, the sampling procedure is very slow and thus impractical for high dimensional data like text. In Bayesian statistics, the *Dirichlet-multinomial allocation* DP_N in [6] has often been applied as a finite approximation to DP (see [6, 9]), which takes on the form

$$G_N = \sum_{l=1}^{N} \pi_l \delta_{\theta_l^*},$$

where $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_N\}$ is an N-vector of probability weights sampled once from a Dirichlet prior $\text{Dir}(\alpha_0/N, \ldots, \alpha_0/N)$, and θ_l^*, $l = 1, \ldots, N$, are i.i.d. sampled from the base distribution G_0. It has been shown that the limiting case of DP_N is DP [6, 9, 12], and more importantly DP_N demonstrates similar stick breaking properties and leads to a similar clustering effect [6]. If N is sufficiently large with

respect to our sample size D, DP_N gives a good approximation to DP.

Under the DP_N model, the plate representation of our model is illustrated in Fig. 2(c). The likelihood of the whole collection $\mathcal{D}$ is

$$\mathcal{L}_{\mathrm{DP}_N}(\mathcal{D}|\alpha_0,\lambda,\beta) = \int_{\boldsymbol{\pi}}\int_{\boldsymbol{\theta}^*} \prod_{d=1}^{D}\Big[\sum_{c_d=1}^{N} p(\mathbf{w}_d|\boldsymbol{\theta}^*,c_d;\beta) \\ p(c_d|\boldsymbol{\pi})\Big]\, dP(\boldsymbol{\theta}^*;G_0)\, dP(\boldsymbol{\pi};\alpha_0) \quad (6)$$

where c_d is the indicator variable saying which unique value θ_l^* document d takes on. The likelihood of document d is therefore written as

$$p(\mathbf{w}_d|\boldsymbol{\theta}^*,c_d;\beta) = \prod_{n=1}^{N_d}\sum_{z_{d,n}=1}^{k} p(w_{d,n}|z_{d,n};\beta)p(z_{d,n}|\theta^*_{c_d}).$$

2.4 Connections to PLSA and LDA

From the application point of view, PLSA and LDA both aim to discover the latent dimensions of data with the emphasis on *indexing*. The proposed Dirichlet enhanced semantic analysis retains the strengths of PLSA and LDA, and further explores the clustering structure of data. The model is a generalization of LDA. If we let $\alpha_0 \to \infty$, the model becomes identical to LDA, since the sampled G becomes identical to the finite Dirichlet base distribution G_0. This extreme case makes documents mutually independent given G_0, since θ_d are i.i.d. sampled from G_0. If G_0 itself is not sufficiently expressive, the model is not able to capture the dependency between documents. The Dirichlet enhancement elegantly solves this problem. With a moderate α_0, the model allows G to deviate away from G_0, giving modelling flexibilities to explore the richer structure of data. The exchangeability may not exist within the whole collection, but between groups of documents with respective atoms θ_l^* sampled from G_0. On the other hand, the increased flexibility does not lead to overfitting, because inference and learning are done in a Bayesian setting, averaging over the number of mixture components and the states of the latent variables.

3 Inference and Learning

In this section we consider model inference and learning based on the DP_N model. As seen from Fig. 2(c), the inference needs to calculate the *a posteriori* joint distribution of latent variables $p(\boldsymbol{\pi},\boldsymbol{\theta}^*,\mathbf{c},\mathbf{z}|\mathcal{D},\alpha_0,\lambda,\beta)$, which requires to compute Eq. (6). This integral is however analytically infeasible. A straightforward Gibbs sampling method can be derived, but it turns out to be very slow and inapplicable to high dimensional data like text, since for each word we have to sample a latent variable z. Therefore in this section we suggest efficient *variational* inference.

3.1 Variational Inference

The idea of variational mean-field inference is to propose a joint distribution $Q(\boldsymbol{\pi},\boldsymbol{\theta}^*,\mathbf{c},\mathbf{z})$ conditioned on some free parameters, and then enforce Q to approximate the *a posteriori* distributions of interests by minimizing the KL-divergence $D_{KL}(Q\|p(\boldsymbol{\pi},\boldsymbol{\theta}^*,\mathbf{c},\mathbf{z}|\mathcal{D},\alpha_0,\lambda,\beta))$ with respect to those free parameters. We propose a variational distribution Q over latent variables as the following

$$Q(\boldsymbol{\pi},\boldsymbol{\theta}^*,\mathbf{c},\mathbf{z}|\boldsymbol{\eta},\boldsymbol{\gamma},\boldsymbol{\psi},\boldsymbol{\phi}) = Q(\boldsymbol{\pi}|\boldsymbol{\eta})\cdot \\ \prod_{l=1}^{N} Q(\theta_l^*|\gamma_l)\prod_{d=1}^{D} Q(c_d|\psi_d)\prod_{d=1}^{D}\prod_{n=1}^{N_d} Q(z_{d,n}|\phi_{d,n}) \quad (7)$$

where $\boldsymbol{\eta},\boldsymbol{\gamma},\boldsymbol{\psi},\boldsymbol{\phi}$ are *variational parameters*, each tailoring the variational *a posteriori* distribution to each latent variable. In particular, $\boldsymbol{\eta}$ specifies an N-dimensional Dirichlet distribution for $\boldsymbol{\pi}$, γ_l specifies a k-dimensional Dirichlet distribution for distinct θ_l^*, ψ_d specifies an N-dimensional multinomial for the indicator c_d of document d, and $\phi_{d,n}$ specifies a k-dimensional multinomial over latent topics for word $w_{d,n}$. It turns out that the minimization of the KL-divergence is equivalent to the maximization of a lower bound of the $\ln p(\mathcal{D}|\alpha_0,\lambda,\beta)$ derived by applying Jensen's inequality [10]. Please see the Appendix for details of the derivation. The lower bound is then given as

$$\mathcal{L}_Q(\mathcal{D}) = \sum_{d=1}^{D}\sum_{n=1}^{N_d}\mathbb{E}_Q[\ln p(w_{d,n}|z_{d,n},\beta)p(z_{d,n}|\boldsymbol{\theta}^*,c_d)] \\ + \mathbb{E}_Q[\ln p(\boldsymbol{\pi}|\alpha_0)] + \sum_{d=1}^{D}\mathbb{E}_Q[\ln p(c_d|\boldsymbol{\pi})] \quad (8) \\ + \sum_{l=1}^{N}\mathbb{E}_Q[\ln p(\theta_l^*|G_0)] - \mathbb{E}_Q[\ln Q(\boldsymbol{\pi},\boldsymbol{\theta}^*,\mathbf{c},\mathbf{z})].$$

The optimum is found setting the partial derivatives with respect to each variational parameter to be zero,

which gives rise to the following updates

$$\phi_{d,n,i} \propto \beta_{i,w_{d,n}} \exp\Big\{ \sum_{l=1}^{N} \psi_{d,l} \big[\Psi(\gamma_{l,i}) - \Psi(\sum_{j=1}^{k} \gamma_{l,j})\big] \Big\} \tag{9}$$

$$\psi_{d,l} \propto \exp\Big\{ \sum_{i=1}^{k} \Big[\Big(\Psi(\gamma_{l,i}) - \Psi(\sum_{j=1}^{k} \gamma_{l,j})\Big) \sum_{n=1}^{N_d} \phi_{d,n,i} \Big] + \Psi(\eta_l) - \Psi(\sum_{j=1}^{N} \eta_j) \Big\} \tag{10}$$

$$\gamma_{l,i} = \sum_{d=1}^{D} \sum_{n=1}^{N_d} \psi_{d,l} \phi_{d,n,i} + \lambda_i \tag{11}$$

$$\eta_l = \sum_{d=1}^{D} \psi_{d,l} + \frac{\alpha_0}{N} \tag{12}$$

where $\Psi(\cdot)$ is the digamma function, the first derivative of the log Gamma function. Some details of the derivation of these formula can be found in Appendix. We find that the updates are quite interpretable. For example, in Eq. (9) $\phi_{d,n,i}$ is the *a posteriori* probability of latent topic i given one word $w_{d,n}$. It is determined both by the corresponding entry in the β matrix that can be seen as a *likelihood* term, and by the possibility that document d selects topic i, i.e., the *prior* term. Here the prior is itself a weighted average of different θ_l^* s to which d is assigned. In Eq. (12) η_l is the *a posteriori* weight of π_l, and turns out to be the tradeoff between empirical responses at θ_l^* and the prior specified by α_0. Finally since the parameters are coupled, the variational inference is done by iteratively performing Eq. (9) to Eq. (12) until convergence.

3.2 Parameter Estimation

Following the empirical Bayesian framework, we can estimate the hyper parameters α_0, λ, and β by iteratively maximizing the lower bound $\mathcal{L}_Q$ both with respect to the variational parameters (as described by Eq. (9)-Eq. (12)) and the model parameters, holding the remaining parameters fixed. This iterative procedure is also referred to as variational EM [10]. It is easy to derive the update for β:

$$\beta_{i,j} \propto \sum_{d=1}^{D} \sum_{n=1}^{N_d} \phi_{d,n,i} \delta_j(w_{d,n}) \tag{13}$$

where $\delta_j(w_{d,n}) = 1$ if $w_{d,n} = j$, and 0 otherwise. For the remaining parameters, let's first write down the parts of $\mathcal{L}$ in Eq. (8) involving α_0 and λ:

$$\mathcal{L}_{[\alpha_0]} = \ln\Gamma(\alpha_0) - N \ln\Gamma\big(\frac{\alpha_0}{N}\big) + \big(\frac{\alpha_0}{N} - 1\big) \sum_{l=1}^{N} \big[\Psi(\eta_l) - \Psi(\sum_{j=1}^{N} \eta_j)\big],$$

$$\mathcal{L}_{[\lambda]} = \sum_{l=1}^{N} \Big\{ \ln\Gamma(\sum_{i=1}^{k} \lambda_i) - \sum_{i=1}^{k} \ln\Gamma(\lambda_i) + \sum_{i=1}^{k} (\lambda_i - 1) \big[\Psi(\gamma_{l,i}) - \Psi(\sum_{j=1}^{k} \gamma_{l,j})\big] \Big\}.$$

Estimates for α_0 and λ are found by maximization of these objective functions using standard methods like Newton-Raphson method as suggested in [3].

4 Empirical Study

4.1 Toy Data

We first apply the model on a toy problem with $k = 5$ latent topics and a dictionary containing 200 words. The assumed probabilities of generating words from topics, i.e. the parameters β, are illustrated in Fig. 3(d), in which each colored line corresponds to a topic and assigns non-zero probabilities to a subset of words. For each run we generate data with the following steps: (1) one cluster number M is chosen between 5 and 12; (2) generate M document clusters, each of which is defined by a combination of topics; (3) generate each document d, $d = 1, \ldots, 100$, by first randomly selecting a cluster and then generating 40 words according to the corresponding topic combinations. For DP_N we select $N = 100$ and we aim to examine the performance for discovering the latent topics and the document clustering structure.

In Fig. 3(a)-(c) we illustrate the process of clustering documents over EM iterations with a run containing 6 document clusters. In Fig. 3(a), we show the initial random assignment $\psi_{d,l}$ of each document d to a cluster l. After one EM step documents begin to accumulate to a reduced number of clusters (Fig. 3(b)), and converge to exactly 6 clusters after 5 steps (Fig. 3(c)). The learned word distribution of topics β is shown in Fig. 3(e) and is very similar to the true distribution.

By varying M, the true number of document clusters, we examine if our model can find the correct M. To determine the number of clusters, we run the variational inference and obtain for each document a weight vector $\psi_{d,l}$ of clusters. Then each document takes the cluster with largest weight as its assignment, and we calculate the cluster number as the number of non-empty clusters. For each setting of M from 5 to 12, we randomize the data for 20 trials and obtain the curve in Fig. 3(f)

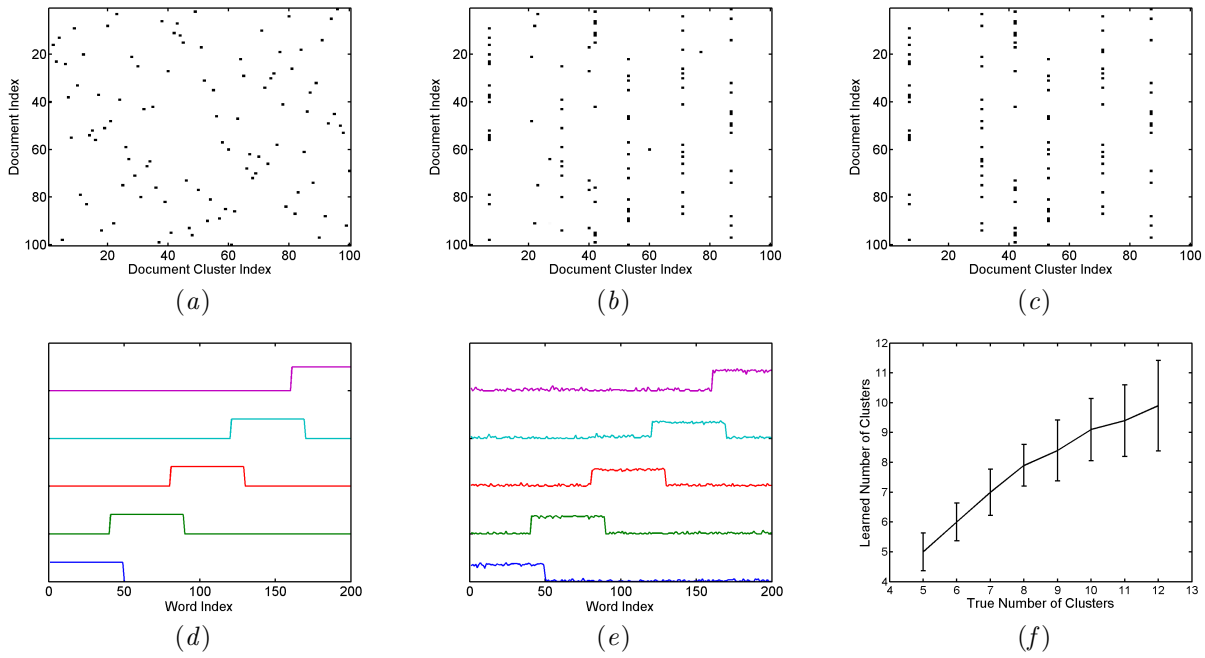


Figure 3: Experimental results for the toy problem. (a)-(c) show the document-cluster assignments $\psi_{d,l}$ over the variational inference for a run with 6 document clusters: (a) Initial random assignments; (b) Assignments after one iteration; (c) Assignments after five iterations (final). The multinomial parameter matrix β of true values and estimated values are given in (d) and (e), respectively. Each line gives the probabilities of generating the 200 words, with wave mountains for high probabilities. (f) shows the learned number of clusters with respect to the true number with mean and error bar.

which shows the average performance and the variance. In 37% of the runs we get perfect results, and in another 43% runs the learned values only deviate from the truth by one. However, we also find that the model tends to get slightly fewer than M clusters when M is large. The reason might be that, only 100 documents are not sufficient for learning a large number M of clusters.

4.2 Document Modelling

We compare the proposed model with PLSI and LDA on two text data sets. The first one is a subset of the Reuters-21578 data set which contains 3000 documents and 20334 words. The second one is taken from the 20-newsgroup data set and has 2000 documents with 8014 words. The comparison metric is *perplexity*, conventionally used in language modelling. For a test document set, it is formally defined as

$$\text{Perplexity}(\mathcal{D}_{\text{test}}) = \exp\left(-\ln p(\mathcal{D}_{\text{test}}) / \sum_d |\mathbf{w}_d|\right).$$

We follow the formula in [3] to calculate the perplexity for PLSI. In our algorithm N is set to be the number of training documents. Fig. 4(a) and (b) show the comparison results with different number k of latent topics. Our model outperforms LDA and PLSI in all the runs, which indicates that the flexibility introduced by DP enhancement does not produce overfitting and results in a better generalization performance.

4.3 Clustering

In our last experiment we demonstrate that our approach is suitable to find relevant document clusters. We select four categories, *autos, motorcycles, baseball* and *hockey* from the 20-newsgroups data set with 446 documents in each topic. Fig. 4(c) illustrates one clustering result, in which we set topic number $k = 5$ and found 6 document clusters. In the figure the documents are indexed according to their true category labels, so we can clearly see that the result is quite meaningful. Documents from one category show similar membership to the learned clusters, and different categories can be distinguished very easily. The first two categories are not clearly separated because they are both talking about vehicles and share many terms, while the rest of the categories, baseball and hockey, are ideally detected.

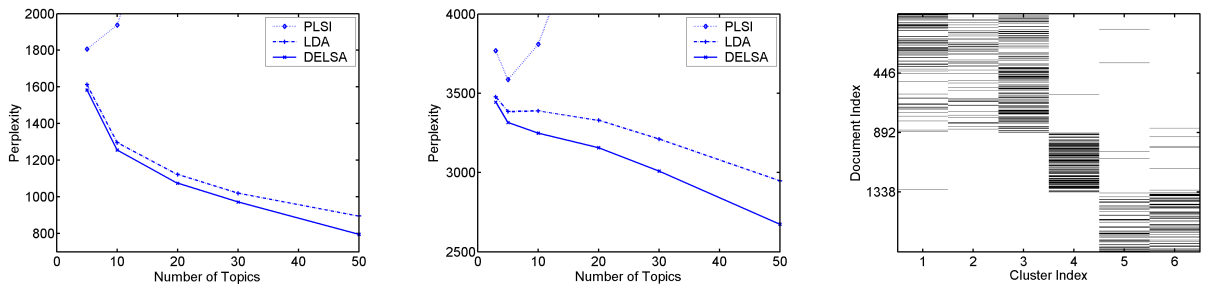


Figure 4: (a) and (b): Perplexity results on Reuters-21578 and 20-newsgroups for DELSA, PLSI and LDA; (c): Clustering result on 20-newsgroups dataset.

5 Conclusions and Future Work

This paper proposes a Dirichlet enhanced latent semantic analysis model for analyzing co-occurrence data like text, which retains the strength of previous approaches to find latent topics, and further introduces additional modelling flexibilities to uncover the clustering structure of data. For inference and learning, we adopt a variational mean-field approximation based on a finite alternative of DP. Experiments are performed on a toy data set and two text data sets. The experiments show that our model can discover both the latent semantics and meaningful clustering structures.

In addition to our approach, alternative methods for approximate inference in DP have been proposed using expectation propagation (EP) [11] or variational methods [16, 2]. Our approach is most similar to the work of Blei and Jordan [2] who applied mean-field approximation for the inference in DP based on a truncated DP (TDP). Their approach was formulated in context of general exponential-family mixture models [2]. Conceptually, DP_N appears to be simpler than TDP in the sense that the *a posteriori* of G is a symmetric Dirichlet while TDP ends up with a generalized Dirichlet (see [8]). In another sense, TDP seems to be a tighter approximation to DP. Future work will include a comparison of the various DP approximations.

Acknowledgements

The authors thank the anonymous reviewers for their valuable comments. Shipeng Yu gratefully acknowledges the support through a Siemens scholarship.

References

[1] M. J. Beal, Z. Ghahramani, and C. E. Rasmussen. The infinite hidden Markov model. In *Advances in Neural Information Processing Systems (NIPS) 14*, 2002.

[2] D. M. Blei and M. I. Jordan. Variational methods for the Dirichlet process. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

[3] D. M. Blei, A. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[4] M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430), June 1995.

[5] T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1:209–230, 1973.

[6] P. J. Green and S. Richardson. Modelling heterogeneity with and without the Dirichlet process. unpublished paper, 2000.

[7] T. Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual ACM SIGIR Conference*, pages 50–57, Berkeley, California, August 1999.

[8] H. Ishwaran and L. F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453):161–173, 2001.

[9] H. Ishwaran and M. Zarepour. Exact and approximate sum-representations for the Dirichlet process. *Can. J. Statist*, 30:269–283, 2002.

[10] M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[11] T. Minka and Z. Ghahramani. Expectation propagation for infinite mixtures. In *NIPS'03 Workshop on Nonparametric Bayesian Methods and Infinite Models*, 2003.

[12] R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal*

of Computational and Graphical Statistics, 9:249–265, 2000.

[13] C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of gaussian process experts. In *Advances in Neural Information Processing Systems 14*, 2002.

[14] J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

[15] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. Technical Report 653, Department of Statistics, University of California, Berkeley, 2004.

[16] V. Tresp and K. Yu. An introduction to nonparametric hierarchical bayesian modelling with a focus on multi-agent learning. In *Proceedings of the Hamilton Summer School on Switching and Learning in Feedback Systems*. Lecture Notes in Computing Science, 2004.

[17] K. Yu, V. Tresp, and S. Yu. A nonparametric hierarchical Bayesian framework for information filtering. In *Proceedings of 27th Annual International ACM SIGIR Conference*, 2004.

Appendix

To simplify the notation, we denote $\boldsymbol{\Xi}$ for all the latent variables $\{\boldsymbol{\pi}, \boldsymbol{\theta}^*, \mathbf{c}, \mathbf{z}\}$. With the variational form Eq. (7), we apply Jensen's inequality to the likelihood Eq. (6) and obtain

$$\begin{aligned}
&\ln p(\mathcal{D}|\alpha_0, \lambda, \beta) \\
&\quad = \ln \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\theta}^*} \sum_{\mathbf{c}} \sum_{\mathbf{z}} p(\mathcal{D}, \boldsymbol{\Xi}|\alpha_0, \lambda, \beta) d\boldsymbol{\theta}^* d\boldsymbol{\pi} \\
&\quad = \ln \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\theta}^*} \sum_{\mathbf{c}} \sum_{\mathbf{z}} \frac{Q(\boldsymbol{\Xi}) p(\mathcal{D}, \boldsymbol{\Xi}|\alpha_0, \lambda, \beta)}{Q(\boldsymbol{\Xi})} d\boldsymbol{\theta}^* d\boldsymbol{\pi} \\
&\quad \geq \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\theta}^*} \sum_{\mathbf{c}} \sum_{\mathbf{z}} Q(\boldsymbol{\Xi}) \ln p(\mathcal{D}, \boldsymbol{\Xi}|\alpha_0, \lambda, \beta) d\boldsymbol{\theta}^* d\boldsymbol{\pi} \\
&\qquad - \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\theta}^*} \sum_{\mathbf{c}} \sum_{\mathbf{z}} Q(\boldsymbol{\Xi}) \ln Q(\boldsymbol{\Xi}) d\boldsymbol{\theta}^* d\boldsymbol{\pi} \\
&\quad = \mathbb{E}_Q[\ln p(\mathcal{D}, \boldsymbol{\Xi}|\alpha_0, \lambda, \beta)] - \mathbb{E}_Q[\ln Q(\boldsymbol{\Xi})],
\end{aligned}$$

which results in Eq. (8).

To write out each term in Eq. (8) explicitly, we have, for the first term,

$$\sum_{d=1}^{D} \sum_{n=1}^{N_d} \mathbb{E}_Q[\ln p(w_{d,n}|z_{d,n}, \beta)] = \sum_{d=1}^{D} \sum_{n=1}^{N_d} \sum_{i=1}^{k} \phi_{d,n,i} \ln \beta_{i,\nu},$$

where ν is the index of word $w_{d,n}$.

The other terms can be derived as follows:

$$\begin{aligned}
&\sum_{d=1}^{D} \sum_{n=1}^{N_d} \mathbb{E}_Q[\ln p(z_{d,n}|\boldsymbol{\theta}^*, c_d)] = \\
&\qquad \sum_{d=1}^{D} \sum_{n=1}^{N_d} \sum_{i=1}^{k} \sum_{l=1}^{N} \psi_{d,l} \phi_{d,n,i} \big[\Psi(\gamma_{l,i}) - \Psi(\sum_{j=1}^{k} \gamma_{l,j})\big], \\
&\mathbb{E}_Q[\ln p(\boldsymbol{\pi}|\alpha_0)] = \ln \Gamma(\alpha_0) - N \ln \Gamma(\frac{\alpha_0}{N}) \\
&\qquad + \Big(\frac{\alpha_0}{N} - 1\Big) \sum_{l=1}^{N} \big[\Psi(\eta_l) - \Psi(\sum_{j=1}^{N} \eta_j)\big], \\
&\sum_{d=1}^{D} \mathbb{E}_Q[\ln p(c_d|\boldsymbol{\pi})] = \sum_{d=1}^{D} \sum_{l=1}^{N} \psi_{d,l} \big[\Psi(\eta_l) - \Psi(\sum_{j=1}^{N} \eta_j)\big], \\
&\sum_{l=1}^{N} \mathbb{E}_Q[\ln p(\theta_l^*|G_0)] = \sum_{l=1}^{N} \Big\{ \ln \Gamma(\sum_{i=1}^{k} \lambda_i) - \sum_{i=1}^{k} \ln \Gamma(\lambda_i) \\
&\qquad + \sum_{i=1}^{k} (\lambda_i - 1) \big[\Psi(\gamma_{l,i}) - \Psi(\sum_{j=1}^{k} \gamma_{l,j})\big] \Big\}, \\
&\mathbb{E}_Q[\ln Q(\boldsymbol{\pi}, \boldsymbol{\theta}^*, \mathbf{c}, \mathbf{z})] = \ln \Gamma(\sum_{l=1}^{N} \eta_l) - \sum_{l=1}^{N} \ln \Gamma(\eta_l) \\
&\quad + \sum_{l=1}^{N} (\eta_l - 1) \big[\Psi(\eta_l) - \Psi(\sum_{j=1}^{N} \eta_j)\big] \\
&\quad + \sum_{l=1}^{N} \Big\{ \ln \Gamma(\sum_{i=1}^{k} \gamma_{l,i}) - \sum_{i=1}^{k} \ln \Gamma(\gamma_{l,i}) \\
&\qquad + \sum_{i=1}^{k} (\gamma_{l,i} - 1) \big[\Psi(\gamma_{l,i}) - \Psi(\sum_{j=1}^{k} \gamma_{l,j})\big] \Big\} \\
&\quad + \sum_{d=1}^{D} \sum_{l=1}^{N} \psi_{d,l} \ln \psi_{d,l} + \sum_{d=1}^{D} \sum_{n=1}^{N_d} \sum_{i=1}^{k} \phi_{d,n,i} \ln \phi_{d,n,i}.
\end{aligned}$$

Differentiating the lower bound with respect to different latent variables gives the variational E-step in Eq. (9) to Eq. (12). M-step can also be obtained by considering the lower bound with respect to β, λ and α_0.

Gaussian Quadrature Based Expectation Propagation

Onno Zoeter **Tom Heskes**
Faculty of Science, University of Nijmegen
o.zoeter@science.ru.nl *tomh@cs.ru.nl*

Abstract

We present a general approximation method for Bayesian inference problems. The method is based on Expectation Propagation (EP). Projection steps in the EP iteration that cannot be done analytically are done using Gaussian quadrature. By identifying a general form in the projections, the only quadrature rules that are required are for exponential family weight functions. The corresponding cumulant and moment generating functions can then be used to automatically derive the necessary quadrature rules. In this article the approach is restricted to approximating families that factorize to a product of one-dimensional families.

The final algorithm has interesting similarities with particle filtering algorithms. We discuss these, and also discuss the relationship with variational Bayes and Laplace propagation. Experimental results are given for an interesting model from mathematical finance.

1 INTRODUCTION

Expectation Propagation (EP) [12] is a powerful deterministic approximate inference technique. It is briefly introduced in Section 3. In EP an initial approximation is iteratively refined by introducing interactions from the exact model and subsequently projecting the extended approximation back onto a chosen parametric family. These projections boil down to a matching of expected sufficient statistics. One of the problems that can stand in the way of a direct application of the EP technique, is that the required integrals implied by the computation of the expected sufficient statistics cannot be done analytically.

It is very natural to try to combine EP with an existing technique to approximate relatively low dimensional integrals. In [18] EP is combined with Laplace approximations. Here we expand upon [10] and [21] and define a general way of combining EP with Gaussian quadrature. Section 6 shows how every projection can be interpreted in a general form. Section 8 describes how, using Stieltjes procedure to construct orthogonal polynomials, the required quadrature rules can be derived automatically. This means that the entire quadrature EP routine could be computer generated for many chosen approximating families. Of course, making use of properties of specific exponential family forms can result in efficiency gains. And as it now stands, the chosen approximating family is required to factorize onto a product of one-dimensional families. However, the procedure as described in this article can form the basis for a method as general as the variational (mean-field) Bayes approach [1, 2].

To facilitate the description of the methods we first introduce our running example and briefly describe EP, the exponential family and Gaussian quadrature. Readers familiar with these basics may briefly glance at Figure 1 and jump to Section 6 right away.

2 STOCHASTIC VOLATILITY MODELS

Many of the, by now classic, results in mathematical finance assume that stocks follow a geometric Brownian motion. This model implies that equidistant log returns are independently, identically and normally distributed. Although the geometric Brownian motion gives a rough description of stock market behavior, the log returns tend to have fatter tails than a normal distribution and do not seem to be homoskedastic. This has led to the development of models where the standard deviation of the log returns, referred to as volatility in finance, is treated as a random variable itself. In this article we study the stochastic volatility

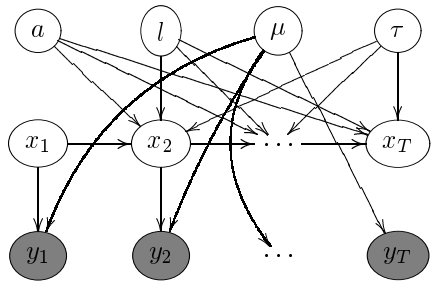


Figure 1: The dynamic Bayesian network that encodes the conditional independences in the stochastic volatility model. Shading emphasizes the fact that a particular variable is observed.

model from [7].

The model is defined in discrete time. The time-index $t = 1, 2, \ldots T$ ranges over equidistant points in time. We define $y_t = \log \frac{S_t}{S_{t-1}}$, the log return at t of stock S. As mentioned above, if the volatility would be constant, the log returns are independently, identically, and normally distributed. We keep the mean of their distributions fixed at μ, but treat the volatility as a random variable itself. The log of the volatility at t is denoted by x_t. The log volatility follows a mean reverting AR(1) process. The complete model reads

$$\begin{aligned} x_t &= a(x_t - l) + l + \epsilon_t, \quad \epsilon_t \sim N(0, \tau^{-1}) \ , \quad (1) \\ y_t &= \mathrm{e}^{x_t}\eta_t + \mu, \quad \eta_t \sim N(0,1) \ . \quad (2) \end{aligned}$$

In the above $N(m, v)$ denotes the Gaussian probability distribution with mean m and variance v. All disturbances ϵ_t and η_t are assumed to be independently drawn.

We will consider factorized subjective priors of the form

$$\begin{aligned} p(a) &= Beta(\alpha_a, \beta_a) \\ p(l) &= N(m_l, v_l) \\ p(\tau) &= Gamma(\alpha_\tau, \beta_\tau) \\ p(\mu) &= N(m_\mu, v_\mu) \\ p(x_1) &= N(m_{x_1}, v_{x_1}) \ . \end{aligned}$$

The full model is depicted as a dynamic Bayesian network [15] in Figure 1. The notational conventions for the various exponential families are introduced in Section 4.

The interest is in smoothed posteriors: $p(x_t|s_{1:T})$, with $t < T$, and in posteriors over the log volatility and drift parameters $p(\theta|s_{1:T})$, with $\theta = \{a, l, \tau, \mu\}$. Unfortunately, as in most Bayesian inference problems, the required integrals cannot be solved analytically.

3 EXPECTATION PROPAGATION

We assume that the joint over all variables in our model factorizes as a product of factors Ψ_t. For the stochastic volatility model this becomes:

$$\begin{aligned} p(a, l, \tau, \mu, x_{1:T}, y_{1:T}) &= \prod \Psi_t(h_t), \quad \text{with} \\ \Psi_1(h_1) &\equiv p(a)p(l)p(\tau)p(\mu)p(x_1) \times \\ &\quad p(y_1|x_1, \mu)p(y_2|x_2, \mu) \times \\ &\quad p(x_2|x_1, a, l, \tau) \ , \\ \Psi_t(h_t) &\equiv p(y_{t+1}|x_{t+1}, \mu) \times \\ &\quad p(x_{t+1}|x_t, a, l, \tau) \ , \end{aligned}$$

for $t = 1, 2, \ldots, T-1$. We denote the hidden variables in the domain of Ψ_t jointly as h_t.

The required posteriors are proportional to this joint

$$p(h|y_{1:T}) \propto \prod \Psi_t(h_t) \ . \quad (3)$$

To derive an expectation propagation algorithm we choose a tractable approximating exponential family $\mathcal{Q}$. For the stochastic volatility model we take a fully factorized approximation with every marginal in the same exponential family as its corresponding prior. I.e. every element $q(h) \in \mathcal{Q}$ satisfies

$$q(h) = q_a(a)q_l(l)q_\tau(\tau)q_\mu(\mu)\prod_{t=1}^{T} q_t(x_t) \ , \quad (4)$$

with $q_a, q_l, q_\tau, q_\mu, q_t$, a Beta, Normal, Gamma, Normal and Normal distribution respectively.

The exact posterior in (3) is approximated by a product of approximate factors $\tilde{\Psi}_t$:

$$p(h|y_{1:T}) \approx q(h) = \prod_{t=1}^{T-1} \tilde{\Psi}_t(h_t) \ . \quad (5)$$

The product is restricted to be in the chosen approximating family $q(h)$. Since this family is taken fully factorized the individual terms can be written without loss of generality as a product of terms over the individual variables in its domain:

$$\begin{aligned} \tilde{\Psi}_t(h_t) &\equiv m_{t\to a}(a)m_{t\to l}(l)m_{t\to\tau}(\tau) \times \\ &\quad m_{t\to\mu}(\mu)m_{t\to x_t}(x_t)m_{t\to x_{t+1}}(x_{t+1}). \end{aligned}$$

The final algorithm can be interpreted as a message passing algorithm, we will therefore refer to the $m_{t\to\cdot}$ terms as the messages going out from factor Ψ_t.

By definition the approximation of a marginal over h_i is now the product of all the messages from factors coming into h_i. E.g.

$$q(a) = \prod_{t=2}^{T-1} m_{t\to a}(a) \ .$$

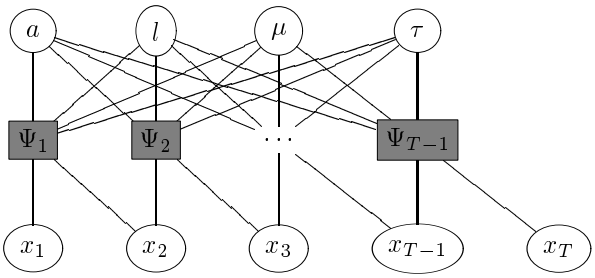


Figure 2: The factor graph corresponding to the choice of factors for the stochastic volatility model. Shaded squares represent factors.

Figure 2 gives a factor graph [8] interpretation of the chosen approximation.

To find an approximation in the family $q(h)$ that is close to the exact posterior (3) EP proceeds as follows:

1. Initialize the approximate factors $\tilde{\Psi}_t$ (and hence the outgoing messages).

2. Compute the initial approximation $q(h)$ from the product of the approximating factors:

$$q(h) = \prod_{t=1}^{T-1} \tilde{\Psi}_t(h_t) \; . \tag{6}$$

   Here we assume without loss of generality that the $\tilde{\Psi}_t$ are initialized such that the initial approximation (6) is normalized.

3. Until all $\tilde{\Psi}_t$ converge:
   (a) Choose a $\tilde{\Psi}_t$ to refine.
   (b) Remove $\tilde{\Psi}_t(h_t)$ from the approximation $q(h)$ by division:

$$q^{\setminus t}(h_t) = \frac{q(h_t)}{\tilde{\Psi}_t(h_t)} \; .$$

   (c) Combine $q^{\setminus t}(h_t)$ with the exact factor $\Psi_t(h_t)$:

$$\tilde{p}(h_t) = \frac{\Psi_t(h_t) q^{\setminus t}(h_t)}{Z_t} \; .$$

   The normalizing constant is defined as $Z_t \equiv \int \Psi_t(h_t) q^{\setminus t}(h_t) dh_t$.
   (d) Since $\tilde{p}(h_t)$ is not in chosen family $\mathcal{Q}$, project:

$$q^{\text{new}}(h_t) = \underset{q \in \mathcal{Q}}{\operatorname{argmin}} \, \mathrm{KL}\left(\tilde{p}(h_t) || q(h_t)\right) \; . \tag{7}$$

   (e) Infer the new approximating factor (and hence messages) by division:

$$\tilde{\Psi}_t^{\text{new}}(h_t) = \frac{q^{\text{new}}(h_t)}{q^{\setminus t}(h_t)} \; .$$

4. Use the normalizing constant of $q(h)$ as an approximation of $p(y_{1:T})$:

$$p(y_{1:T}) \approx \prod_{t=1}^{T-1} Z_t \; .$$

This algorithm is closely related to loopy belief propagation [13]. Just as for loopy belief propagation convergence is not guaranteed.

4 EXPONENTIAL FAMILY MODELS

The approximating family $\mathcal{Q}$ is restricted to be in the exponential family. The exponential family has some pleasant properties: an exponential family is closed under product and division, and the minimum of (7) is determined by a finite number of statistics from $\tilde{p}(h)$. In this section we introduce our notation for exponential family models and give the basic results that are required in the rest of the text.

Exponential family models can be represented as

$$p(y|\theta) = \mathrm{e}^{\theta^\top u(y) - \phi(\theta)} \; . \tag{8}$$

We say that a class $\mathcal{F}$ of models is an exponential family if all its members can be written in the form (8). We refer to θ as the vector of natural parameters or canonical parameters, to $u(y)$ as the vector of sufficient statistics, and to $\phi(\theta)$ as the log partition function. We assume that the family is represented minimally, i.e. that no elements of $u(y)$ are linear combinations of others. From the exponential form in (8) we immediately see that the family is closed under product and division.

The expected values of the sufficient statistics $\langle u(y) \rangle_{p(y|\theta)}$, will be important in our further description of exponential family models. We will refer to them as *natural* moments to contrast them with $\langle y^i \rangle_{p(y|\theta)}$, the i-th moment of $p(y)$. The so-called link-function $g(\cdot)$ maps canonical parameters to natural moments

$$g(\theta) \equiv \langle u(y) \rangle_{p(y|\theta)} = \int u(y) p(y|\theta) dy \; .$$

The link function can also be derived as the first derivative of $\phi(\theta)$ [9]

$$\frac{\partial \phi(\theta)}{\partial \theta} = g(\theta) \; .$$

The KL minimization in the EP projection step (7) boils down to a matching of the natural moments [9]. I.e. $q^{\text{new}}(h_t) = \mathrm{e}^{\gamma^\top u(h_t)}$, with $\gamma = g^{-1}(\langle u(h_t) \rangle_{\tilde{p}(h_t)})$, is the distribution in $\mathcal{Q}$ that matches the natural moments of the distribution $\tilde{p}(h_t)$.

5 GAUSSIAN QUADRATURE

Gaussian quadrature is a general technique to approximate integrals of the form $\int_a^b f(x)K(x)dx$, where $K(x)$ is a known non-negative function. In the inference algorithms $K(x)$ will be a (normalized) exponential family distribution, not necessarily Gaussian (the method is due to Gauss, which explains the name of the quadrature procedure).

Based on $K(x)$, n points $\mathcal{X}_1, \ldots, \mathcal{X}_n$ and n corresponding weights $w_1, \ldots, w_n$ are chosen such that

$$\int_a^b K(x)f(x)dx \approx \sum_{i=1}^{n} f(\mathcal{X}_i)w_i \ ,$$

is exact if $f(x)$ is a polynomial of degree at most $2n-1$. General procedures to determine $\mathcal{X}_i$ and w_i are based on sets of n polynomials which are orthogonal w.r.t. $K(x)$ and the interval $[a, b]$. See e.g. [17] for an introduction to Gaussian quadrature.

To approximate multi-dimensional integrals over factoring weight functions, grids can be used. The location of the grid points can be determined from the position of the points determined for the individual marginal weight functions. The weights are simply multiplied:

$$\begin{aligned}\int_a^b \int_c^d f(x,y)K_x(x)K_y(y)dxdy \\ \approx \int_a^b \sum_i w_i f(\mathcal{X}_i, y)K_y(y)dy \\ \approx \sum_j \sum_i w_j w_i f(\mathcal{X}_i, \mathcal{Y}_i) \ .\end{aligned}$$

Throughout this paper we will assume that elements in $\mathcal{Q}$ factorize as products of univariate marginals. More advanced rules derived directly from the exactness of integrals over multinomials form an interesting extension.

6 QUADRATURE EP

If we combine the EP steps (3.b) to (3.d), we can identify a general way of using Gaussian quadrature as a numerical projection method. Combining the steps, an update from q to $\tilde{p}$ is defined as:

$$\tilde{p}(h_t) \equiv \frac{\Psi_t(h_t)}{Z_t \tilde{\Psi}_t(h_t)} q(h_t) \ , \tag{9}$$

with $Z_t \equiv \int \frac{\Psi_t(h_t)}{\tilde{\Psi}_t(h_t)} q(h_t)dh_t$. If we identify $q(h_t)$ as the weight function, we can approximate the normalization constant using Gaussian quadrature. Since the weight function is by construction a normalized exponential family distribution, we can make use of this fact in deriving the quadrature rules.

Integrals involving $\tilde{p}(h)$ can now be approximated by reweighted function evaluations. To project $\tilde{p}$ back onto the chosen family, we first approximate the natural moments $\langle u(y) \rangle_{\tilde{p}}$ using the reweighted points. Then, the inverse of the link function is used to find the parameters of q^{new} given the approximate moments.

Summarizing, quadrature EP updates an approximation of factor Ψ_t as follows:

1. Compute weights w_i and points $\mathcal{X}_i$ for the (factorized) old posterior $q(h_t)$.

2. Reweight every point:

$$\tilde{w}_i = \frac{w_i \frac{\Psi_t(\mathcal{X}_i)}{\tilde{\Psi}_t(\mathcal{X}_i)}}{\sum_j w_j \frac{\Psi_t(\mathcal{X}_j)}{\tilde{\Psi}_t(\mathcal{X}_j)}} \ . \tag{10}$$

3. Approximate the natural moments using the reweighted points

$$\langle u(h_t) \rangle_{\tilde{p}} \approx \sum_i \tilde{w}_i u(\mathcal{X}_i) \ .$$

4. The parameters for the new approximation are found by inverting the link function:

$$\begin{aligned} q^{\text{new}}(h_t) &= \mathrm{e}^{\nu_t^\top u(h_t)}, \quad \text{with,} \\ \nu_t &= g^{-1}(\sum_i \tilde{w}_i u(\mathcal{X}_i)) \ . \end{aligned}$$

5. Infer new messages by division

$$\tilde{\Psi}_t^{\text{new}}(h_t) = \frac{q^{\text{new}}(h_t)}{q^{\setminus t}(h_t)} = \frac{q^{\text{new}}(h_t)}{q(h_t)} \tilde{\Psi}_t(h_t) \ .$$

Just as for EP itself, the iterative refinement of factors is not guaranteed to converge.

There is a strong resemblance between the above algorithm and particle filtering algorithms (see e.g. [4]). Just as in the particle filtering algorithm, points are used twice: once to approximate a normalization constant and once to approximate a posterior distribution. Also, if q^{new} has its center of mass in a very different area from q, only a few points $\mathcal{X}_i$ will get non-negligible weight. As a result the approximation of q^{new} is very poor. We compare the algorithm in more detail to particle filtering and other approaches in Section 9.

7 APPROXIMATE INFERENCE IN THE SV MODEL

7.1 QUADRATURE EP FOR THE SV MODEL

The specific quadrature EP algorithm for the stochastic volatility model only depends on our choice of the approximating family $\mathcal{Q}$. We will assume that elements in $\mathcal{Q}$ factorize as a product of univariate marginals. For the stochastic volatility model, $\mathcal{Q}$ consists of all elements of the form (4).

To complete the general description of Section 6, we need to define how the points and weights are chosen for $q(h)$, and how new parameters are found given approximate moments.

For the Gaussian components $q_l(l), q_\mu(\mu), q_t(x_t)$, the points and weights can be determined using Gauss-Hermite polynomials. See e.g. [17] for a description.

Since

$$\int_{-\infty}^{\infty} f(x)N(x|m,v)dx = \int_{-\infty}^{\infty} f\left(y\sqrt{v}+m\right)N(y|0,1)dy\,, \tag{11}$$

we can determine points once for $N(x|0,1)$, and transform these when we encounter an integral for $N(x|m,v)$ with $m \neq 0$ or $v \neq 1$.

In exponential form we write the normal distribution as

$$\begin{aligned} N(x|m,v) &= \frac{1}{\sqrt{2\pi v}}\mathrm{e}^{\frac{-(x-m)^2}{2v}} \\ &= \mathrm{e}^{\theta_N^\top u_N(x)-\phi_N(\theta_N)}, \text{with} \\ \theta_N &\equiv \begin{bmatrix} h \equiv \frac{m}{v} \\ L \equiv -\frac{1}{2v} \end{bmatrix} \\ u_N(x) &= \begin{bmatrix} x \\ x^2 \end{bmatrix} \\ \phi_N(\theta_N) &= \frac{1}{2}\log\left(-\frac{\pi}{L}\right) - \frac{h^2}{4L}\,. \end{aligned}$$

The link function is

$$g_N(\theta_N) = \frac{\partial\phi_N(\theta_N)}{\partial\theta_N} = \begin{bmatrix} -\frac{h}{2L} & -\frac{1}{2L} - \frac{h^2}{4L^4} \end{bmatrix}^\top .$$

which we can invert analytically

$$g_N^{-1}(\langle u_N(u)\rangle) = \begin{bmatrix} \frac{\langle x\rangle}{\langle x^2\rangle - \langle x\rangle^2} & \frac{-1}{2\left(\langle x^2\rangle - \langle x\rangle^2\right)} \end{bmatrix}^\top .$$

Unfortunately, there is no result analogous to (11) for the Beta and Gamma distributions. We can, however, rewrite integrals involving Beta and Gamma distributions as integrals over some well-studied weight functions.

We write the Beta distribution as

$$\begin{aligned} Beta(x|\alpha,\beta) &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1} \\ &= \mathrm{e}^{\theta_B^\top u_B(x)-\phi_B(\theta_B)}, \text{with} \\ \theta_B &\equiv \begin{bmatrix} \alpha-1 \\ \beta-1 \end{bmatrix} \\ u_B(x) &= \begin{bmatrix} \log x \\ \log(1-x) \end{bmatrix} \\ \phi_B(\theta_B) &= \log\frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}\,. \end{aligned}$$

Any integral with a Beta weight function can be transformed to a Gauss-Jacobi form as follows

$$\begin{aligned} &\int_0^1 f(x)\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1}dx \\ &= c(\alpha,\beta)\int_{-1}^{1} f\left(\frac{y+1}{2}\right)(1+y)^{\alpha-1}(1-y)^{\beta-1}dy\,. \end{aligned}$$

with $c(\alpha,\beta) = \frac{1}{2^{\alpha+\beta-1}}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}$. Just as for the Gauss-Hermite case, the coefficients of the polynomials are known functions of the parameters in the weight function, and very good approximations of the roots exist [17].

The link function is given by

$$g_B(\theta_B) = \frac{\partial\phi_B(\theta_B)}{\partial\theta_B} = \begin{bmatrix} \psi(\alpha)-\psi(\alpha+\beta) \\ \psi(\beta)-\psi(\alpha+\beta) \end{bmatrix}^\top ,$$

where $\psi(x) \equiv \frac{\partial}{\partial x}\Gamma(x)$, is the digamma function. There exists no analytical inverse of this link function (in fact, depending on definitions, the link function itself is not analytic due to the digamma function). Minimizing the squared distance $\left(g_B(\theta_B) - \langle u_B(y)\rangle_{\tilde{p}}\right)^\top \left(g_B(\theta_B) - \langle u_B(y)\rangle_{\tilde{p}}\right)$ w.r.t. θ_B gives a numerical inverse.

The Gamma distribution is given by

$$\begin{aligned} Gamma(x|\alpha,\beta) &= \frac{\beta^\alpha}{\Gamma(\alpha)}x^{\alpha-1}\mathrm{e}^{-\beta x} \\ &= \mathrm{e}^{\theta_G^\top u_G(x)-\phi_G(\theta_G)}, \text{with} \\ \theta_G &\equiv \begin{bmatrix} -\beta \\ \alpha-1 \end{bmatrix} \\ u_G(x) &= \begin{bmatrix} x \\ \log x \end{bmatrix} \\ \phi_G(\theta_G) &= \log\frac{\Gamma(\alpha)}{\beta^\alpha}\,. \end{aligned}$$

We can rewrite Gamma integrals into a Gauss-Laguerre form by noting that

$$\begin{aligned} &\int_0^\infty f(x)\frac{\beta^\alpha}{\Gamma(\alpha)}x^{\alpha-1}\mathrm{e}^{-\beta x}dx \\ &= \frac{1}{\Gamma(\alpha)}\int_0^\infty f\left(\frac{y}{\beta}\right)y^{\alpha-1}\mathrm{e}^{-y}dy \end{aligned}$$

The link function is

$$g_G(\theta_G) = \frac{\partial \phi_G(\theta_G)}{\partial \theta_G} = \begin{bmatrix} \frac{\alpha}{\beta} \\ \psi(\alpha) - \log \beta \end{bmatrix}^\top .$$

Which also has no analytic inverse.

There is a dynamical aspect in the model, so it seems most logical to update the approximate factors $\tilde{\Psi}_t$ in a forward-backward fashion. The next section presents a useful initialization of the approximation $q(h)$ and of the messages. This completes the description of the quadrature EP algorithm for the stochastic volatility model. Section 7.3 gives results of experiments.

7.2 A FIRST FORWARD PASS

In principle, many initializations of $q(h)$ and $\tilde{\Psi}_t$ will do. Although, as mentioned in Section 6 we want to take care that $q(h)$ 'has mass' wherever $q^{\text{new}}(h)$ has. Otherwise q^{new}, and hence its sufficient statistics, are poorly approximated by points computed from $q(h)$.

We can initialize $q(h)$ dynamically by constructing $q(h_t)$ during the first forward pass. We initialize $q_a(a), q_l(l), q_\tau(\tau), q_\mu(\mu)$, and $q_1(x_1)$ with the priors from the model. All messages are initialized as 1. The consecutive $q_{t+1}(x_{t+1})$ are initialized by drawing points from $q(a, l, \tau, x_t)$ and propagating these through the deterministic part of the transition model (1). These propagated points are used to construct a Gaussian approximation of $q_{t+1}(x_{t+1})$. To introduce the stochastic part of (1) we simply add the corresponding Gaussian disturbance to the preliminary estimate of $q_{t+1}(x_{t+1})$. Now a regular EP update can be performed.

7.3 EXPERIMENTS

The procedure from Section 6 and the initialization scheme from Section 7.2 allow us to compute approximate posteriors of the form (4). Figure 3 presents approximate posteriors over a, l, $v \equiv \tau^{-1}$, and μ for a very small artificially generated five-slice problem. The solid curves show posteriors computed using quadrature EP. The histograms present approximations based on 100.000 Gibbs samples. Since the problem is so small, we expect the Gibbs approximation to be near exact. Despite the restrictive form of the approximating family (4), the EP approximation is reasonable. Code for the stochastic volatility model is available from `www.snn.ru.nl/~orzoeter`.

Figure 4 demonstrates the risk of an ill-matched initial estimate of $q(h)$. As mentioned in Section 6, if the initial estimate of $q(h)$ has low weight in a significant part of $p(h|y_{1:T})$ the resulting approximation is poor. The example is constructed such that the

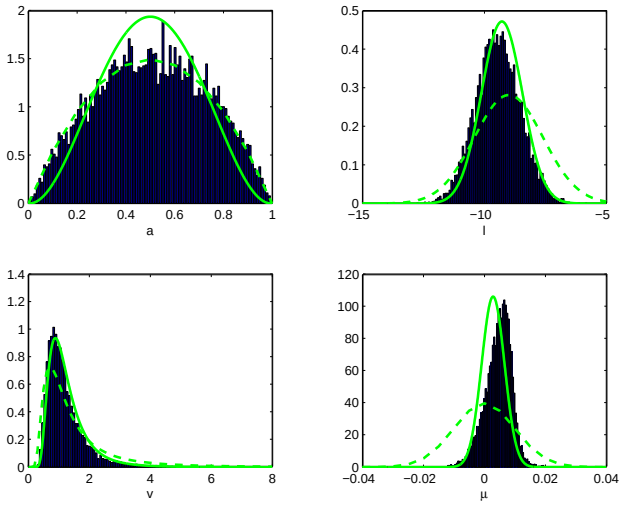


Figure 3: Approximate posteriors for a small problem with five observations. A Gibbs approximation is presented by the histograms, the solid curves show a quadrature EP approximation, the dashed curves are the subjective priors.

prior $p(v)$ (represented by a dotted line) and posterior $p(v|y_{1:T})$ (approximated by Gibbs samples in the histogram) have their mass in different areas. If $q(v)$ is initialized with the prior, the quadrature points and weights that are drawn from $p(v)$ (presented as circles) result in a poor approximation of $p(v|y_{1:T})$ (represented by a solid line). A related problem would have occurred if the prior for μ would have been very flat in the example in Figure 3. The exact posterior is very peaked close to zero. The quadrature points from a very flat distribution centered at zero would have one point at zero which effectively takes all the weight in (10). The result would be an approximation of $p(v|y_{1:T})$ by a delta-peak.

Simple experiments seem to be encouraging. However, the strong influence of the initial estimate $q(h)$ requires extra care. Either better ways of initializing $q(h)$, a different proposal distribution, or smart adaptive ways to position the quadrature points are needed to construct a reliable approximation. Note that particle filters suffer from related problems.

8 COMPUTER GENERATED RULES

Ideally we would like to have a very general class of approximation techniques where code for specific models can easily be computer generated. BUGS [19] is a very successful example for Gibbs sampling, and VIBES [20] for mean-field based approaches.

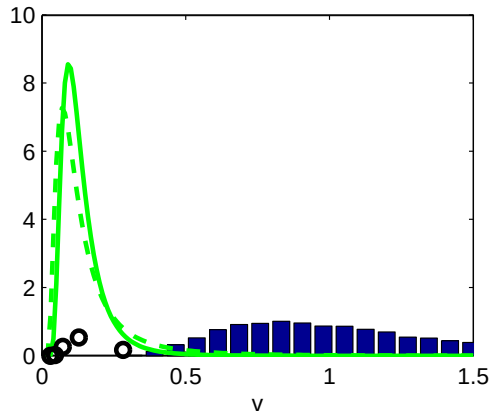


Figure 4: A quadrature EP approximation of $p(v|y_{1:T})$ based on an ill-matched initial approximation $q(h)$. See text for details.

The general form (9) of the quadrature EP algorithm ensures that the weight functions are always (products of) exponential family models. There are several techniques of recursively constructing orthogonal polynomials that only require the evaluation of moments. Stieltjes procedure [17] is among the best known, but perhaps not the most stable.

If the moment generating function

$$M(t) = \int \mathrm{e}^{tx} p(x) dx \ ,$$

of an exponential distribution is known, the computation of the moments follows from differentiating the moment generating function:

$$\left\langle x^i \right\rangle_{p(x)} = \left. \frac{d^i M(t)}{dt^i} \right|_0 \ .$$

This is mechanical and can be automated. Extensions to multi-dimensional quadrature rules is a topic for future research.

9 RELATED METHODS

In this section we give a, by no means complete, comparison to related methods.

Perhaps the method closest to the one presented here is particle filtering [4]. Instead of determining points using Gaussian quadrature, particle filtering algorithms draw points from a proposal distribution. And, instead of using reweighted points to project the posterior onto a chosen parametric form, the reweighted points (particles) are kept as a non-parametric approximation of the posterior. Quadrature based filters in the fully Gaussian case are a.o. described in [16] and [3]. Tangent to our approach are lattice particle filters [14] that stay within the particle filter class, but generate proposal points in a clever way.

In Laplace propagation [18] the KL projection in the EP algorithm is replaced by a Laplace approximation. This may form a good alternative in many settings, especially if there is a relatively large number of observations and posteriors are well approximated by Gaussians.

Note that, for the current model, any approximation method that approximates x_t and y_t jointly as a Gaussian, will result in very poor results. Since x_t and y_t are uncorrelated in (2), a Gaussian approximation will treat x_t and y_t as independent. Hence a prior for x_t will not be updated in the light of observing y_t. The unscented Kalman filter [6] will therefore, for this model, only propagate the prior, i.e. break down completely. See [21] for more details.

EP and mean-field approaches are closely related. However they are not as closely related as the factored form (4) of $\mathcal{Q}$ may lead us to assume. Both methods can be derived starting from the following variational objective

$$-\log p(y_{1:T}) = \min_{q \in \mathcal{P}} -\log p(y_{1:T}) + \mathrm{KL}\left(q(h) || p(h|y_{1:T})\right) \ . \tag{12}$$

If $\mathcal{P}$ is the set of all valid distributions on h the KL term in (12) vanishes at the minimum, and the equality is indeed an equality. Both mean-field and EP approaches arrive at an approximation by replacing $\mathcal{P}$ with a tractable set. For mean-field approaches $\mathcal{P}$ is replaced by the set $\mathcal{Q}$ with factored elements (4). Perhaps confusingly, the EP approach is not based on the same set $\mathcal{Q}$. Instead of replacing $\mathcal{P}$ by a set of simpler, but proper distributions q, the minimization is over sets of overlapping pseudo marginals with certain constistency constraints (see e.g. [11, 5] for more details). The choice of $\mathcal{Q}$ as a family on which $\tilde{p}$ is projected, determines the overlaps of these pseudo marginals. Since the approximation retains more structure of the original model, the hope is that the approximation is better than a fully factorized approximation of $\mathcal{P}$.

10 DISCUSSION

We have shown how general quadrature approximations can be identified in the standard EP scheme. The approach appears to be rather flexible and is closely related to particle filtering algorithms. The projections onto a chosen family allows iterative improvements of approximations. This is in contrast to particle filtering algorithms that can select the position of points (particles) only once and can only reweight in the light of extra information.

The running time of the quadrature EP approach is exponential in the number of variables in h_t. This is because we have approximated integrals over h_t by a grid over all variables in h_t. This complexity is identical to Kikuchi and junction tree algorithms in fully discrete networks. More advanced quadrature rules may result in a running time sub-exponential in the largest clique size. The cost of determining the grid points depends on the particular choice of $\mathcal{Q}$, the exponential family on which the posterior is projected. For Gaussians, grid points can be computed once and rescaled whenever needed (11). In the worst case, a set of orthogonal polynomials has to be constructed, of which the roots must be found numerically.

When the quadrature based method is computationally too intensive, replacing steps 1 to 3 from the algorithm in Section 6 by importance sampling may form an interesting alternative.

The Beta and Gamma components in the choice of $\mathcal{Q}$ in Section 7 imply extra computational effort. After seeing many observations, the posterior over θ will tend to be Gaussian. It is interesting to establish for what observation sizes the extra effort is worthwhile.

The procedure to generate orthogonal polynomials and quadrature rules described in this article is among the best studied in the literature. But it is unlikely that it is the optimal one for the EP framework. We would at least require rules for multi-dimensional weight functions, taking the posterior of parameters factorized is probably relatively coarse. Also, traditional Gaussian quadrature is designed to achieve zero error for a class of polynomials. For the current application it may be interesting to require good performance for different classes of functions.

Acknowledgments

We would like to thank Alexander Ypma, Wim Wiegerinck, Tjeerd Dijkstra and Ali Taylan Cemgil for helpful discussions. The Gibbs approximations were produced using WinBUGS which is available from `www.mrc-bsu.cam.ac.uk/bugs`.

References

[1] H. Attias. Inferring parameters and structure of latent variable models by variational Bayes. In *Proceedings UAI*, 1999.

[2] M. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, University College London, 2003.

[3] E. Bolviken and G. Storvik. Deterministic and stochastic particle filters in state space models. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag.

[4] A. Doucet, N. D. Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer–Verlag, 2001.

[5] T. Heskes and O. Zoeter. Generalized belief propagation for approximate inference in hybrid Bayesian networks. In *Proceedings AISTATS*, 2003.

[6] S. Julier and J. K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Proceedings of AeroSense*, 1997.

[7] S. Kim, N. Shephard, and S. Chib. Stochastic volatilty: Likelihood inference and comparison with ARCH models. *Review of Economic Studies*, 65, 1998.

[8] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2), 2001.

[9] S. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[10] U. Lerner. *Hybrid Bayesian Networks for Reasoning about Complex Systems*. PhD thesis, Stanford University, 2002.

[11] T. Minka. The EP energy function and minimization schemes. Technical report, 2001.

[12] T. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings UAI*, 2001.

[13] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings UAI*, 1999.

[14] D. Ormoneit, C. Lemieux, and D. J. Fleet. Lattice particle filters. In *Proceedings UAI*, 2001.

[15] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan-Kaufmann, 1988.

[16] A. Pole and M. West. Efficient Bayesian learning in non-linear dynamic models. *Journal of Forecasting*, 9(2), 1990.

[17] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Programming*. Cambridge University Press, 2nd edition, 1992.

[18] A. Smola, V. Vishwanathan, and E. Eskin. Laplace propagation. In *Proceedings NIPS*. 2004.

[19] D. J. Spiegelhalter, A. Thomas, and N. G. Best. Computation on Bayesian graphical models. In *Bayesian Statistics 5*, pages 407–425, 1996.

[20] J. Winn and C. Bishop. Structured variational distributions in VIBES. In *Proceedings AISTATS*, 2003.

[21] O. Zoeter, A. Ypma, and T. Heskes. Improved unscented Kalman smoothing for stock volatility estimation. In *Proceedings MLSP*, 2004.