Learning to Parse Images

by

Yee Whye Teh

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Learning to Parse Images

Yee Whye Teh

Master of Science

Graduate Department of Computer Science

University of Toronto

2000

Segmentation and recognition are important subtasks of image interpretation. The general approach to statistical image interpretation tries to solve each separately. This introduces various problems as segmentation and recognition are interrelated.

These problems can be avoided by solving segmentation and recognition simultaneously. This is achieved by viewing segmentation and recognition as subtasks of finding the correct parse tree of the image, and viewing image interpretation as a search in the space of parse trees.

Credibility networks are an instantiation of this idea. They are graphical models which describe a probability distribution over all possible parse trees with the leaves corresponding to pixels.

The parameters of a credibility network can be learned and inference can be achieved using mean field approximations. During inference, the results of segmentation and recognition are iteratively improved upon. Simulations showed that credibility networks can perform interesting toy problems, for example hand-written digit classification and segmentation.

Acknowledgements

This research has been carried out with funding from NSERC and ITRC through my supervisor Geoffrey Hinton. I was also supported by a University of Toronto Fellowship during the 1997/1998 academic year and an Ontario Graduate Scholarship during the 1998/1999 academic year.

The general idea of using graphical models to parse images started with discussions among Geoff Hinton, Zoubin Ghahramani and Chris Williams.

I would like to thank Geoff for his guidance, inspiration and patience. Many apologies to Geoff too for the months and months of delay before I finally submitted this thesis. I would also like to thank members of the Neural Networks group at the University of Toronto : Andy, Alberto, Brian and Zoubin, for answering my never ending stream of questions. Further thanks to Zoubin for being my second reader. I wish to thank Fahiem Bacchus for being my unofficial supervisor during my last year as an undergraduate at the University of Waterloo and for getting me into research in artificial intelligence.

On the personal side, there have been many people who have touched and shaped my life. I am deeply in debt to them because they are the meaning and joy of my life. First and foremost I wish to thank my family. I have spent most of my life since the age of twelve away from my family and have missed them dearly. My love to Mei for being with me. I would also like to thank Mr. Loh Kon, my computer science teacher in high school, who have introduced me to computer science and nurtured me during my years in Singapore. Many thanks to Chiew Farn, whom I have known since thirteen and who has grown up with me in Singapore. I value his companionship dearly. And of course, my '2493' friends from Bronte College : Sze Yong, Tan Cheng, Hooi Beng, Huey Jen, Jenny, Chee Kwan, Lip Jin, Carol, Jacky, Wee Haw, Ah Leong and my elder brother Yee Neng. Those days in Bronte College were the best.

Dedication

To Grandma

Contents

1 Introduction **1**

1.1 Image Interpretation 1

1.2 Thesis Organization 4

1.3 Statistical Models 4

1.3.1 Sigmoid Belief Networks 4

1.3.2 Multiscale Models 5

1.3.3 TRAFFIC 5

1.3.4 Mixtures of Factor Analyzers 6

1.4 Graphical Models 6

1.5 The EM Algorithm 7

1.5.1 Variational Methods 8

1.5.2 Monte Carlo Methods 10

2 Credibility Networks **13**

2.1 A New Approach to Image Interpretation 13

2.2 Binary Credibility Networks 14

2.3 Mean Field Approximations 17

2.3.1 Integrating Out Λ 18

2.3.2 A More Complex Q 19

2.3.3 Approximations 21

2.4 Relation to Some Models 21

2.4.1 Hierarchical Community of Experts 21

2.4.2 Cooperation Versus Competition 23

3 Experimental Data **25**

3.1 The Bars Problem 25

3.2 Classifying Hand-written Digits 29

3.3 Segmenting Hand-written Digits 31

3.4 The Faces Problem 34

3.5 Correcting The Faces Problem 41

4 Conclusions **44**

4.1 Discussion 44

4.2 Future Work 46

Bibliography **48**

List of Tables

3.1 Confusion matrix of ML classifier. The ij^{th} entry is the number of times an image of digit i was classified as digit j. 30

3.2 a) Confusion matrix for single digit classification. The ij^{th} entry is the number of times an image of digit i was classified as digit j. b) Percentage of cases inferred correctly by the trained network for each combination of two digits. c) Similarly for the author. 32

List of Figures

2.1 A parse tree embedded in a DAG. The top unit is the "grandparent" unit. Dashed edges are edges of the DAG while edges of the parse tree are in bold. To avoid clutter we did not draw the edges connecting the top unit to units in the lower layers. 14

2.2 A unit in a binary credibility network. Each unit i and its parenthood unit i^* accept connections from its potential parents $j \in pa(i)$, and sends connections to its children and their parenthood units. 15

3.1 Sample images from the bars problem. 25

3.2 Hidden to output weights. 26

3.3 Encoding mutual exclusion between horizontal bars and vertical bars. . . 27

3.4 Encoding mutual exclusion between n units. 28

3.5 a) Sample images of digits used to train and test the ML classifier. b) Sample images generated by the trained models. 29

3.6 Sample images from the test set. The classes of digits of the images in a row are given to the left. 31

3.7 Segmentations of pairs of digits. In each group, the top two images are the original images. The middle two images are both superpositions of the top two images. The bottom two images are the segmented images produced by the credibility network. 33

3.8 Some images from the faces problem. 34

3.9 Image reconstructions of SBN. The left image in each group is the prediction made by the SBN, while the right image is the actual training image. The area of each pixel is the probability that the pixel is on. 35
3.10 Images generated by SBN trained on the faces problem. The large diagrams are averages over the randomly generated images, with a large black pixel meaning that the pixel is on for almost all of the images generated. The eight small diagrams below each large diagram are samples from the randomly generated images. 36
3.11 Predictions of the credibility network (left) and the actual images (right). 37
3.12 Images generated by credibility network trained on the faces problem. The large diagrams are averages over the randomly generated images, with a large black pixel meaning that the pixel is on for almost all of the images generated. The eight small diagrams below each large diagram are samples from the randomly generated images. 39
3.13 Hidden to output weights for some low level units. 40
3.14 An example of how low level units interact in a complicated way. 40
3.15 Sample images. 41
3.16 Each image is the average of 1000 images generated from the model when only the top level units given above each image are on. 42
3.17 Hidden to output weights for all low level units. 42

Chapter 1

Introduction

1.1 Image Interpretation

What does it mean to interpret an image?

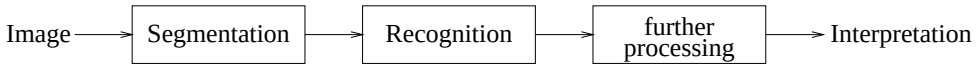
Perhaps the most obvious answer is to recognize the objects present in the image. The task of recognition has been the focus of attention of statistical pattern recognition for the past 40 years. The paradigm problem is to classify an object from a vector of features extracted from the image. With the advent of backpropagation [34], the choice of features and the choice of weights to put on these features became part of a single, overall optimization and impressive performance was obtained for restricted but important tasks such as hand-written character identification [7].

A significant weakness of many current recognition systems is their assumption that images contain exactly one object to be recognized. Often, further assumptions are made that the images are normalized. Except in highly controlled situations, this is obviously not true. To deal with this, a separate preprocessing phase is often performed where images are segmented and normalized such that each segment contains one object in a standard view.

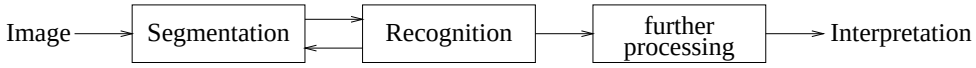
There are various problems with such an image interpretation system, in which segmentation is first carried out followed by recognition. Firstly since we are segmenting the image in order to aid recognition, we have little prior knowledge as to what the objects are, or their locations in the image. It is very hard to perform segmentation accurately

with such scarce knowledge. Hence researchers have to rely on hand-crafted, ad hoc heuristics which work well only in specific circumstances. Secondly, by segmenting an image, we remove the object to be recognized from the context in which it arises. Although this helps in removing the clutter present in the rest of the image, it might also reduce the ability to recognize an object correctly because the context in which an object arises gives a great deal of information about the nature of the object[1]. Thirdly, each object can be described in terms of its parts, which can also be viewed as objects on their own. Then the question of how fine-grained the segmentations should be arises. In the words of David Marr : "Is a nose an object? Is a head one? ... What about a man on a horseback?" [28].

The first problem stems from the fact that many current systems use a purely bottom-up approach, as depicted by the flow chart below.

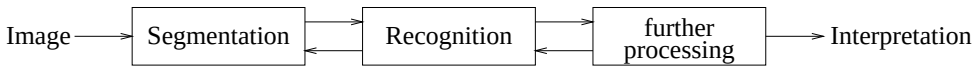


By introducing more knowledge of what is in the image, we can improve our segmentation process. An intuitive solution for this is to couple the segmentation and recognition phases, as shown below.



The idea is to first obtain a rough segmentation of the image, and pass the segments to the recognition phase, obtaining a rough idea of what objects are present in the image. Now using the partial results from the recognition phase, we can improve the initial segmentation. This is repeated until the algorithm converges to a stable set of segments and objects.

The second problem can be handled similarly, if we now couple the segmentation and recognition phases to the higher level processes :



Using initial results from the segmentation and recognition phases, we can obtain partial

[1] Allen Jepson gave this example : a strange shape located at the side of a door can still be recognized as a door knob, even though it might be impossible to determine what it is in isolation.

results from the higher level processes. These results can now act as the context required to improve the performance of the recognition phase and vice versa.

We can use graphical models to actually couple the different phases of image interpretation together. Graphical models have recently came into prominence in various communities including statistics and neural networks. There are a number of reasons for this. Graphical models describe the causal relationships between random variates in an intuitively appealing fashion. Further, graphical models can be fitted to data and the underlying relationships between random variates can be extracted. During inference, graphical models exhibit a phenomenon known as 'explaining away' [33], which is very important for segmentation. Perhaps most importantly for our cause, graphical models allow for the integration of top down information and bottom up evidence in a statistically correct manner. The coupling of the different phases as described above will follow from the algorithms for inference in graphical models (chapter 2).

A possible solution for the problem of how fine-grained segmentations should be is an old idea from structural image analysis: the parse tree of an image. Suppose the image is first segmented into tiny regions, each of which can be described as an edge, a bar, a uniform region etc. Then the tiny regions are grouped together (i.e. the set of regions is segmented). Using the descriptions of the tiny regions, the larger regions can be recognized as longer edges, curves etc. These regions are grouped further into larger and larger objects. As the regions are grouped recursively, we get a hierarchical, tree structured representation of the image. In this tree structure, we can obtain the segmentation results at every granularity level. At each level we can obtain compact descriptions of the regions, e.g. a line, a nose, or a head. This is the parse tree of the image.

This thesis describes work done in the investigation of a new paradigm for statistical image interpretation called credibility networks. Credibility networks are graphical models designed to infer the parse trees of images. In the process of inference, segmentation and recognition at multiple levels of abstraction are performed together using an EM-like algorithm. As segmentation is now part of the overall optimization, the need for hand-crafted, ad hoc segmentation heuristics can be eliminated.

1.2 Thesis Organization

The following section, 1.3, describes popular and recent models used in statistical pattern recognition. The models are briefly described and their advantages and drawbacks are discussed. Then section 1.4 describes graphical models and how to learn and do inference with them, in particular, the EM algorithm is described in section 1.5. Chapter 2 discusses credibility networks in detail. Section 2.2 discusses a simple class of credibility networks we call binary credibility networks that do not deal with instantiation parameters. Section 2.3 derives mean field learning and inference rules for binary credibility networks. Then section 2.4 describes some works in relation to credibility networks. The hierarchical mixtures of experts motivates a way to extend credibility networks to handle instantiation parameters in section 2.4.1. Chapter 3 presents a number of toy experiments on binary credibility networks. Finally chapter 4 describes some problems with the current model and suggests various future endeavors to address the problems as well as to extend the model.

1.3 Statistical Models

1.3.1 Sigmoid Belief Networks

A sigmoid belief network (SBN) is a graphical model consisting of a number of stochastic binary[2] units connected in a directed acyclic graph [29]. The probability of a unit being on is the sigmoid of the weighted sum of the inputs the unit receives from its parents :

$$P(s_i|pa(s_i)) = sigmoid(\sum_{j \in pa(s_i)} w_{ij}s_j) \tag{1.1}$$

where $sigmoid(x) = (1 + \exp(-x))^{-1}$ and $pa(s_i)$ is the set of parents of unit i. In general, as each unit has multiple parents, it is intractable to compute the posterior distribution over hidden variables when certain variables are observed. However, Neal showed that Gibbs sampling can be used effectively for inference [29]. Efficient methods

[2]By binary units we mean the activations take on values of 0 or 1.

of approximating the posterior distribution were introduced later [8, 16, 36, 38] and these approaches were shown to yield good density models for binary images of hand-written digits [11]. The problem with these models which make them inappropriate for modeling images is that they fail to respect the 'single-parent' constraint : in the correct interpretation of an image of opaque objects each object-part belongs to at most one object – images need parse trees, not parse DAGs.

1.3.2 Multiscale Models

Multiscale models [2, 4, 25] are interesting generative models for images that use a fixed tree structure. Nodes high up in the tree control large blocks of the image while bottom level leaves correspond to individual pixels. Because a tree structure is used, it is easy to compute the exact posterior distribution over the latent (non-terminal) nodes given an image. As a result the approach worked much better than Markov Random Fields which generally involve an intractable partition function. A disadvantage is that there are serious block boundary artifacts. Overlapping trees were proposed as solutions by smoothing the transition from one block to another [23]. A more serious disadvantage is that the tree cannot possibly correspond to a parse tree because it is the same for every image.

1.3.3 TRAFFIC

Zemel, Mozer and Hinton [43] proposed a neural network model in which the activities of neurons are used to represent the instantiation parameters of objects or their parts, i.e. the viewpoint-dependent coordinate transformation between an object's intrinsic coordinate system and the image coordinate system. The weights on connections are then used to represent the viewpoint-invariant relationship between the instantiation parameters of the whole object and the instantiation parameters of its parts. This model captures viewpoint invariance nicely and corresponds to the way viewpoint effects are handled in computer graphics, but there was no good inference procedure for hierarchical models and no systematic way of sharing modules that recognize parts of objects among

multiple competing object models.

1.3.4 Mixtures of Factor Analyzers

Simard *et al* [14, 40] noted that small changes in object instantiation parameters result in approximately linear changes in (real-valued) pixel intensities. This means that a linear model like factor analysis can capture these small changes in object instantiation parameters successfully. To model larger changes, many locally linear models can be pieced together. Hinton, Dayan and Revow [17] proposed a mixture of factor analyzers for this. Ghahramani and Hinton [13] showed that the mixture can be learned using an exact EM algorithm. Bishop has recently shown how to make this approach much more computationally efficient [41, 42]. To make the approach really efficient, however, it is necessary to have multiple levels of factor analyzers and to allow an analyzer at one level to be shared by several competing analyzers at the next level up. Deciding which subset of the analyzers at one level should be controlled by one analyzer at the level above is equivalent to image segmentation or the construction of part of a parse tree and the literature contains no proposals on how to achieve this.

1.4 Graphical Models

A graphical model is a probability model represented as a graph. The nodes $i \in I$ of the graph stand for the random variables $X = \{x_i | i \in I\}$ of interest[3], and the edges represent dependencies among the variables. The edges can be directed or undirected (both kinds can appear in the same graph). Here we shall only consider directed acyclic graphs (DAGs), in which case the edges can be interpreted as causations. Further, given the values of its parents, a node is conditionally independent of its other ancestors. More formally, the probability distribution function for X is

$$P(X) = \prod_{i \in I} P(x_i | x_j \text{ for } j \in pa(i)) \tag{1.2}$$

[3]For simplicity in description, we may refer to a node when we meant the associated random variable and vice versa. What we meant should be clear from the context.

Similar decompositions of the probability distribution function of X into local distributions exist for other types of graphical models. This notion of conditional independence is a very important property of graphical models, because it captures the structure of the underlying random process generating the variables and translates that into a concrete mathematical form which we can work with.

Each term in (1.2) is a local probability distribution involving the variable at a node conditioned on the values of variables at its parents. The distribution is parameterized by some parameters $\theta = \{\theta_i | i \in I\}$,

$$P(X|\theta) = \prod_{i \in I} P(x_i|\theta_i, x_j \text{ for } j \in pa(i)) \tag{1.3}$$

Using Bayes' rule given observations $\mathcal{O}$ at some nodes, the posterior distribution over the unobserved variables $\mathcal{H} = X \setminus \mathcal{O}$ is

$$P(\mathcal{H}|\mathcal{O}, \theta) = \frac{P(\mathcal{H}, \mathcal{O}|\theta)}{P(\mathcal{O}|\theta)} = \frac{P(X|\theta)}{P(\mathcal{O}|\theta)} \tag{1.4}$$

The process of computing $P(\mathcal{H}|\mathcal{O})$ is called inference, as we are inferring the values of the unobserved variables given some observations, hence of properties of the unobserved process giving rise to the observations. The quantity $P(\mathcal{O}|\theta)$ is called the likelihood of the observations $\mathcal{O}$. The computation of the likelihood term in graphical models is in general NP-hard [5]. This means that exact inference is often an intractable process and approximations are often required.

Suppose we have some data $\mathcal{D}$ to which we wish to fit a graphical model. We can do this by maximizing the likelihood of generating the data $P(\mathcal{D})$ with respect to θ . This process is called learning. Since log is a strictly increasing function, it is often easier to increase the log likelihood instead.

For more in-depth information on graphical models, the reader is referred to Heckerman [15] and Buntine [3].

1.5 The EM Algorithm

The standard way to fit a graphical model is by the EM algorithm [10]. Suppose we have a model with parameters θ and the values $\mathcal{O}$ at a number of nodes $I_{\mathcal{O}}$ were observed.

Let $I_{\mathcal{H}} = I \setminus I_{\mathcal{O}}$ be the remaining unobserved nodes and $\mathcal{H}$ be the values at these nodes. The EM algorithm is an iterative algorithm which finds maximum likelihood estimates for θ given the observations $\mathcal{O}$.

Initially, estimates are used for $\mathcal{H}$ and θ. For each iteration, the algorithm consists of two steps : an expectation (E) step and a maximization (M) step. During the E step, the distribution over $\mathcal{H}$ is inferred from the observations. That is, the posterior distribution over the hidden variables given the observations $\mathcal{O}$ and θ is computed :

$$Q \leftarrow P(\mathcal{H}|\mathcal{O}, \theta) \tag{1.5}$$

Then, during the M step, the parameters θ of the model are learned to fit the observations, that is, θ is re-estimated to be the maximum likelihood parameters assuming that the distribution found in the E step is correct :

$$\theta^{new} \leftarrow \arg\max_{\theta} \Big\{ E_Q[\log P(\mathcal{O}, \mathcal{H}|\theta)] \Big\} \tag{1.6}$$

Each iteration of the algorithm increases the true likelihood of generating the observations unless a local maximum or a saddle point has already been attained. In fact, if instead of performing a full M step, we update θ such that it only improves $E_Q[\log P(\mathcal{O}, \mathcal{H}|\theta)]$ (e.g. a gradient ascent method) but not maximizes it, the procedure is still guaranteed to increase the true likelihood. Such an M step is called a partial M step and the algorithm is now called the generalized EM (GEM) algorithm.

A partial M step is sometimes necessary because the full M step might involve an intractable or inefficient non-linear optimization. In such a case gradient ascent based methods might still be feasible. Similarly calculating the posterior distribution in the E step might be infeasible, or the posterior distribution itself might be too complex and hence expensive to represent. There are two broad approaches to bypass this problem : variational methods and Monte Carlo methods.

1.5.1 Variational Methods

Neal and Hinton [31] showed that the EM algorithm can be viewed as coordinate descent in a cost function (called the variational free energy) that resembles the free energy in

statistical physics :

$$\mathcal{F}(Q,\theta) = E_Q[-\log P(\mathcal{H},\mathcal{O}|\theta)] - E_Q[-\log Q(\mathcal{H}|\theta)] \tag{1.7}$$

The first expectation in (1.7) is the (variational) energy of the system while the second expectation is the (variational) entropy of the distribution Q. In this formulation, the EM algorithm just alternates between minimizing $\mathcal{F}(Q,\theta)$ with respect to Q and with respect to θ.

It is easy to verify that

$$KL(Q\|P(\mathcal{H}|\mathcal{O},\theta)) = \log P(\mathcal{H}|\mathcal{O},\theta) - (-\mathcal{F}(Q,\theta)) \tag{1.8}$$

where $KL(Q\|P(\mathcal{H}|\mathcal{O},\theta))$ is the Kullback-Leibler (KL) divergence of $P(\mathcal{H}|\mathcal{O},\theta)$ from Q. By the positivity of the KL divergence [6], we see that $-\mathcal{F}(Q,\theta)$ is a lower bound on the log likelihood $\log P(\mathcal{H}|\mathcal{O},\theta)$ with equality exactly when Q equals the posterior distribution $P(\mathcal{H}|\mathcal{O},\theta)$ [24]. This shows that the EM algorithm (as well as a wide range of its variants, including the generalized EM) would always increase the log likelihood unless already at a local maximum or saddle point [31]. Further, it shows that even if we choose a Q that decreases $\mathcal{F}$ but not minimizes it, we are still increasing a lower bound on the log likelihood.

Consider a family of tractable approximating distributions $Q(\mathcal{H}|\theta,\xi)$ over $\mathcal{H}$ parameterized by ξ. The parameters ξ are called variational parameters while Q is called a variational approximation. Instead of working with the correct but intractable posterior distribution $P(\mathcal{H}|\mathcal{O},\theta)$, we work with a feasible approximation $Q(\mathcal{H}|\theta,\xi)$ to it. That is, during the E step we minimize or improve upon (e.g. using some gradient descent method) $\mathcal{F}(Q(\mathcal{H}|\theta,\xi),\theta)$ with respect to ξ. This is the variational method.

The simplest variational approximation is the maximum a posteriori (MAP) approximation. Here we take $Q(\mathcal{H}|\theta,\xi)$ to be the distribution that assigns a probability of 1 to one particular value $\mathcal{H}(\xi)$ of $\mathcal{H}$. The MAP approximation is very simple to use and can be a good approximation if the true posterior has a sharply peaked dominant mode (although finding the sharp peak of the true posterior can be quite hard and the approximation might end up at the maximum of a sub-dominant mode). However, due to it's

simplicity, the MAP approximation can be used as a first attempt at evaluating a new model.

Another simple variational approximation is the mean field (MF) approximation. This approximation assumes that the hidden variables $\mathcal{H} = \{\mathcal{H}_i\}_{i \in I_\mathcal{H}}$ are independent given the observations :

$$Q(\mathcal{H}|\theta, \xi) = \prod_{i \in I_\mathcal{H}} Q_i(\mathcal{H}_i|\theta, \xi_i) \tag{1.9}$$

For real-valued hidden variables, one might use Gaussian distributions for approximation. For binary-valued hidden variables, ξ_i consists of just the mean of the variable $\mathcal{H}_i$. The MF approximation is very good when the graph is very dense and the inter-node influence is relatively weak [32]. The MF approximation has also been shown to be effective in training sigmoid belief networks [36, 37].

Variational approximations introduce a bias towards models whose posterior distributions are close to the simple approximating distributions. On the one hand, this means that the models we get after learning may not be as efficient as possible; on the other, we can think of the variational method as introducing a regularizer that prefers more tractable models. Variational methods are deterministic, and there is always a known lower bound on the log likelihood which never decreases. This is advantageous because guarantees can be made regarding the performance of the model. A disadvantage is that inference is an optimization process which always decreases the free energy locally and can often get stuck in local minimas. An excellent introduction to variational methods is given by Jordan *et al* [24].

1.5.2 Monte Carlo Methods

Instead of trying to approximate the posterior distribution using some predefined parameterized subset of distributions, we can try to sample from the true posterior distribution. Suppose in the E step we get n independent samples $\{\mathcal{H}_1, \ldots, \mathcal{H}_n\}$ from the posterior distribution. In the M step, we can approximate the expectation in (1.6) with

$$E[\log P(\mathcal{H}, \mathcal{O}|\theta)] \approx \frac{1}{n} \sum_{i=1}^{n} \log P(\mathcal{H}_i, \mathcal{O}|\theta) \tag{1.10}$$

and update θ by improving the approximation (1.10) instead. This is the Monte Carlo (MC) method. As the samples are from the posterior distribution, for large enough n, we are guaranteed that the approximation is accurate. Unfortunately many samples may be required. It is also often unclear how many independent samples are sufficient. A survey of existing Monte Carlo methods is given by MacKay [26].

However it is often hard to obtain samples from the posterior distribution directly. Sometimes it is possible to construct an ergodic Markov process whose invariant distribution is the posterior $P(\mathcal{H}|\mathcal{O},\theta)$. In this case, one can run the Markov process, obtaining a sequence of states $\{\mathcal{H}^{(0)},\mathcal{H}^{(1)},\dots\}$. It can be shown that $\mathcal{H}^{(n)}$ is a sample from a distribution $P^{(n)}$ that converges to $P(\mathcal{H}|\mathcal{O},\theta)$ in the KL divergence sense as $n\to\infty$ [30]. We can then take $\mathcal{H}^{(n)}$ to be a sample from the posterior once $P^{(n)}$ is close enough to $P(\mathcal{H}|\mathcal{O},\theta)$. This method of obtaining samples from the posterior is called the Markov chain Monte Carlo (MCMC) method.

A problem for MCMC methods is the time required for $P^{(n)}$ to converge to the posterior distribution. In many cases it is either not known or the time to convergence is too long. Further, as consecutive states are not independent, potentially many states have to be visited in between samples, or we have to run the Markov process multiple times to make sure the samples we take are independent. Details of Markov chain Monte Carlo methods are provided by Neal [30], including ways to improve the convergence times of MCMC methods.

A popular MCMC method is Gibbs sampling [12]. In Gibbs sampling, to generate the current state $\mathcal{H}^{(t)}=\{\mathcal{H}_i^{(t)}\}_{i\in I_\mathcal{H}}$ from the previous state $\mathcal{H}^{(t-1)}=\{\mathcal{H}_i^{(t-1)}\}_{i\in I_\mathcal{H}}$ each hidden unit i is visited according to some order $<$ on $I_\mathcal{H}$, and $\mathcal{H}_i^{(t)}$ is sampled from the marginal distribution $P(\mathcal{H}_i^{(t)}\,|\,\mathcal{O},\mathcal{H}_j^{(t)}\,\forall j<i,\mathcal{H}_k^{(t-1)}\,\forall i<k,\theta)$. The transition distribution is

$$T(\mathcal{H}^{(t)}|\mathcal{H}^{(t-1)})=\prod_{i\in I_H}P(\mathcal{H}_i^{(t)}\,|\,\mathcal{O},\mathcal{H}_j^{(t)}\,\forall j<i,\mathcal{H}_k^{(t-1)}\,\forall i<k,\theta) \tag{1.11}$$

Due to the conditional independence of the nodes of the graphical model, the marginal distribution is only dependent on the values of the Markov blanket of i[4]. This means that Gibbs sampling can be achieved by an algorithm that only uses local information. Thus

[4] These are the parents of i, children of i, and those nodes that share a child with i.

a neural or hardware implementation of Gibbs sampling is possible. Gibbs sampling is popular because in many circumstances, even though the posterior distribution can be quite complicated, the marginal distribution is very simple and easy to sample from.

Chapter 2

Credibility Networks

2.1 A New Approach to Image Interpretation

In this chapter we develop a class of graphical models called credibility networks. In credibility networks the possible interpretations of an image are parse trees, with nodes representing object-parts and containing instantiation parameters. The possible parse trees of an image are constrained to be the spanning subtrees of an underlying directed acyclic graph (DAG), with the leaves being the pixels of an image. To describe the connectivity of the parse tree each node in the DAG has an associated parenthood node which describes the parent of the node in the tree.

We assume there is a "grandparent" unit in the DAG that is a parent of every other unit in the DAG. This serves to make sure the parse tree is connected. It also acts as a default parent for units which do not have any other parents (section 2.2).

To avoid confusion, from now onwards we shall refer to the parent of a node in the parse tree as the "parent" of the node, and the parents of the node in the DAG as its "potential parents" (since these are the possible parents of the node in the tree). Figure 2.1 shows one such tree spanning a DAG that is a multilayered graph.

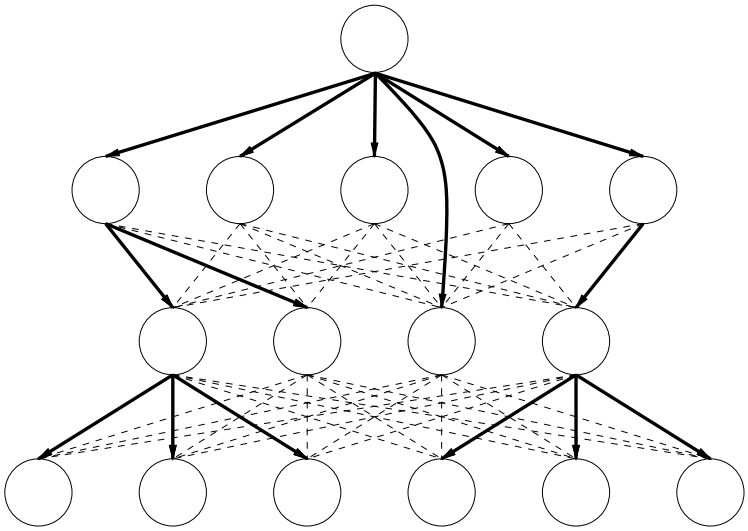


Figure 2.1: A parse tree embedded in a DAG. The top unit is the "grandparent" unit. Dashed edges are edges of the DAG while edges of the parse tree are in bold. To avoid clutter we did not draw the edges connecting the top unit to units in the lower layers.

2.2 Binary Credibility Networks

In this section, we describe in full detail a simplified class of credibility networks in which instantiation parameters of objects are not dealt with. In fact each node in our DAG is binary and represents only the presence or absence of the corresponding object. We shall called these binary credibility networks. The extension to include instantiation parameters will be described in section 2.4.1, although for experimental purposes only the binary credibility networks will be used.

To encode the structure of the parse trees, each node i in the DAG is associated with a parenthood node i^*. In addition to the connections already present in the DAG, there are connections from the potential parents $pa(i)$ of i to i^*, and a connection from i^* to i. This is shown in figure 2.2.

The random variable associated with node i is s_i and is defined as

$$s_i = \begin{cases} 1 & \text{if object } i \text{ is present,} \\ 0 & \text{if object } i \text{ is absent.} \end{cases}$$

The random variable associated with node i^* is $\lambda_i = \{\lambda_{ij}\}_{j \in pa(i)}$. This is a multinomial

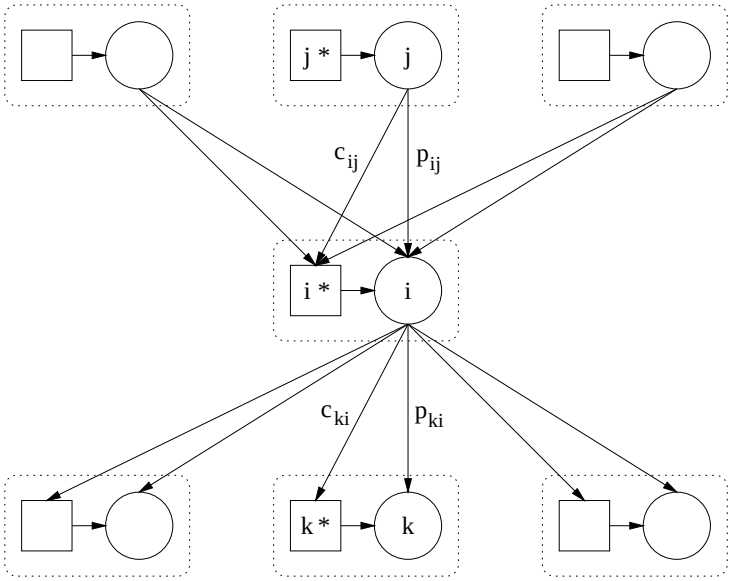


Figure 2.2: A unit in a binary credibility network. Each unit i and its parenthood unit i^* accept connections from its potential parents $j \in pa(i)$, and sends connections to its children and their parenthood units.

random variate representing the unique parent of i in the parse tree of the image :

$$\lambda_{ij} = \begin{cases} 1 & \text{if parent of } i \text{ is } j, \\ 0 & \text{if parent of } i \text{ is not } j. \end{cases}$$

Let $S = \{s_i : i \in I\}$ and $\Lambda = \{\lambda_i : i \in I\}$.

In line with the notion of the conditional independence of the variables at each node of a graphical model, we have

$$P(\Lambda, S) = \prod_{i \in I} P\left(\lambda_i \,\middle|\, s_j \,\forall\, j \in pa(i)\right) P\left(s_i \,\middle|\, \lambda_i, s_j \,\forall\, j \in pa(i)\right) \tag{2.1}$$

Consider $\pi_{ij} = P(\lambda_{ij} = 1 \,|\, s_j \,\forall\, j \in pa(i))$. Let the connections from the parents j of i to i^* have positive weights c_{ij}. Define

$$\pi_{ij} = \frac{c_{ij} s_j}{\sum_{k \in pa(i)} c_{ik} s_k}. \tag{2.2}$$

This says that only objects which are present can be parents of i. This makes sense as we do not want parts belonging to absent objects appearing in the generated images. On

the other hand, it makes sense for parts of objects to be absent. For example, if a cup is absent from the image, we should expect all parts of it to be absent. If a cup handle is present, it does not belong to the absent cup. On the other hand, a cup could be present while the handle is missing (it could be on the occluded side of the cup).

The credibility parameter c_{ij} is an unnormalized prior probability that j is the parent of i. The credibilities are scale invariant, that is, multiplying all incoming credibilities c_{ij} for $j \in pa(i)$ by a fixed constant will not affect the values of the parenthood probabilities π_{ij}. Further the denominator in (2.2) can potentially be 0, when all $s_k = 0$. To rectify both of these problems, we assume there is a "grandparent" unit, $1 \in I$, which is a potential parent of every other unit in the DAG. Unit 1's activation is fixed at 1. Further, the outgoing credibilities of unit 1 are also set at 1 to fix the scales of the other credibilities. Unit 1 can be viewed as a default parent for a unit $i \in I$ when no other unit is available to be i's parent.

Suppose we have now chosen the parent k of i. To determine s_i, all we need is a probability p_{ik} that i is on, given the parent of i is k. Note s_i should not depend on the presence of units in $pa(i)$ which are not the parent of i in the parse tree. This gives

$$\begin{aligned} P(s_i \mid \lambda_i, s_j \,\forall\, j \in pa(i)) &= \prod_{j \in pa(i)} \left(p_{ij}^{s_j}(1-p_{ij})^{1-s_j}\right)^{\lambda_{ij}} \\ &= p_{ik}^{s_k}(1-p_{ik})^{1-s_k} \quad \text{where } \lambda_{ik} = 1 \end{aligned} \tag{2.3}$$

The parameters c_{ij} and p_{ij} are given at relevant edges in figure 2.2. Let $\theta = \{c_{ij}, p_{ij} : i \in I, j \in pa(i)\}$ be the set of parameters. Now the joint distribution becomes

$$P(\Lambda, S|\theta) = \prod_{i \in I} \prod_{j \in pa(i)} \left(\frac{c_{ij}s_j}{\sum_{k \in pa(i)} c_{ik}s_k} p_{ij}^{s_i}(1-p_{ij})^{1-s_i}\right)^{\lambda_{ij}} \tag{2.4}$$

Note that Λ can be integrated out of (2.4) giving

$$P(S|\theta) = \prod_{i \in I} \frac{\sum_{j \in pa(i)} c_{ij}s_j p_{ij}^{s_i}(1-p_{ij})^{1-s_i}}{\sum_{k \in pa(i)} c_{ik}s_k} \tag{2.5}$$

Suppose observations $\mathcal{O} \subset S$ were made at some subset of units $I_{\mathcal{O}} \subset I$. Let $\mathcal{H} = S \setminus \mathcal{O}$

be the unobserved or hidden variables and $I_{\mathcal{H}} = I \setminus I_{\mathcal{O}}$. The likelihood of generating $\mathcal{O}$

$$\begin{aligned} P(\mathcal{O}|\theta) &= \int P(\mathcal{H}, \mathcal{O}|\theta)\, d\mathcal{H} \\ &= \int \prod_{i \in I} \frac{\sum_{j \in pa(i)} c_{ij} s_j p_{ij}^{s_i} (1-p_{ij})^{1-s_i}}{\sum_{k \in pa(i)} c_{ik} s_k}\, d\mathcal{H} \end{aligned} \tag{2.6}$$

cannot be efficiently computed. Hence exact inference is not tractable and approximate inference is required.

2.3 Mean Field Approximations

Consider a naive mean field approximation over $\mathcal{H}$ and Λ, in which the variables are assumed to be completely factorized :

$$Q(\mathcal{H}, \Lambda) = \prod_{i \in I_H} \sigma_i^{s_i} (1-\sigma_i)^{1-s_i} \prod_{i \in I} \prod_{j \in pa(i)} \mu_{ij}^{\lambda_{ij}} \tag{2.7}$$

where σ_i can be viewed as the mean of s_i and μ_{ij} can be viewed as the mean of λ_{ij} under the posterior distribution $P(\mathcal{H}, \Lambda|\mathcal{O}, \theta)$. The variational energy is

$$\mathcal{E} = \int\int -\log \left(\prod_{i \in I} \prod_{j \in pa(i)} \left(\frac{c_{ij} s_j}{\sum_{k \in pa(i)} c_{ik} s_k} p_{ij}^{s_i} (1-p_{ij})^{1-s_i} \right)^{\lambda_{ij}} \right) Q(\mathcal{H}, \Lambda)\, d\Lambda\, d\mathcal{H} \tag{2.8}$$

Suppose there is an i' and a j' such that $\sigma'_j < 1$, $j' \in pa(i')$, and $\mu_{i'j'} > 0$. Pushing the log operator through the multiplications and divisions, consider the log of the $c_{ij}s_j$ term corresponding to i' and j'. This is

$$\begin{aligned} &\int\int \log\left(c_{i'j'} s_{j'}\right)^{\lambda_{i'j'}} \sigma_{j'}^{s_{j'}} (1-\sigma_{j'})^{1-s_{j'}} \mu_{i'j'}^{\lambda_{i'j'}}\, d\lambda_{i'j'}\, ds_{j'} \\ =&\mu_{i'j'} \log c_{i'j'} + \int\int \left(\lambda_{i'j'} \log s_{j'}\right) \sigma_{j'}^{s_{j'}} (1-\sigma_{j'})^{1-s_{j'}} \mu_{i'j'}^{\lambda_{i'j'}}\, d\lambda_{i'j'}\, ds_{j'} \end{aligned} \tag{2.9}$$

The double integral above involves a term $(\log 0)(1-\sigma_{j'})\mu_{i'j'} = -\infty$, so the variational energy in (2.8) is $\mathcal{E} = \infty$.

The energy is infinite because we assigned to a state $(\mathcal{H}, \Lambda)$ a non-zero probability under Q when $(\mathcal{H}, \Lambda)$ has zero probability of being generated by the model. In this case, when $s_{j'} = 0$, the probability under Q of i' choosing j' as it's parent is $\mu_{i'j'} > 0$, but it is impossible to have j' as i''s parent because j' is off.

There are two ways around this problem. One way is to simplify by first integrating out Λ. Another way is to make μ_{ij} be dependent on $\mathcal{H}$ such that $\mu_{ij}(\mathcal{H}) = 0$ whenever $s_j = 0$. Then the probability of an impossible state under Q would still be zero and hence the variational energy would not be infinite.

2.3.1 Integrating Out Λ

Working from (2.5), and using a factored distribution for $\mathcal{H}$:

$$Q_1(\mathcal{H}) = \prod_{i \in I_H} \sigma_i^{s_i} (1 - \sigma_i)^{1 - s_i} \tag{2.10}$$

the variational free energy is

$$\begin{aligned} -\mathcal{F}_1(Q_1, \theta) \;\; = \;\; & \sum_{i \in I} \left(E_{Q_1} \Big[\log \sum_{j \in pa(i)} c_{ij} s_j p_{ij}^{s_i} (1 - p_{ij})^{1 - s_i} - \log \sum_{j \in pa(i)} c_{ij} s_j \Big] \right) - \\ & \sum_{i \in I_H} \Big(\sigma_i \log \sigma_i + (1 - \sigma_i) \log (1 - \sigma_i) \Big) \end{aligned} \tag{2.11}$$

As the grandparent unit 1 is a potential parent of every other unit, and $c_{i1} = 1$ for all i, the summations in (2.11) are always positive, hence the variational free energy is never infinite.

Let $ch(i)$ be the potential children of i (i.e. the children of i in the DAG) and $r_{ij} = c_{ij} p_{ij}^{s_i} (1 - p_{ij})^{1 - s_i}$. While c_{ij} is the unnormalized prior probability of j being the parent of i, r_{ij} can be viewed as the unnormalized posterior probability of j being the parent of i given S. It is the product of the prior probability c_{ij} and the likelihood $p_{ij}^{s_i} (1 - p_{ij})^{1 - s_i}$ of generating s_i. The inference rules are

$$\sigma_i = sigmoid \left(\begin{array}{l} E_{Q_1} \Big[\log \sum_{j \in pa(i)} c_{ij} s_j p_{ij} - \log \sum_{j \in pa(i)} c_{ij} s_j (1 - p_{ij}) \Big] \\ + \sum_{l \in ch(i)} E_{Q_1} \Big[\log \sum_{j \in pa(l)} r_{lj} s_j - \log \sum_{j \in pa(l)} c_{lj} s_j \Big]_{\sigma_i = 0}^{\sigma_i = 1} \end{array} \right) \tag{2.12}$$

Updating σ_i with (2.12) will always increase $-\mathcal{F}_1$.

Let D be the training set and Q_1^d be the mean field approximation to the posterior distribution over $\mathcal{H}$ given the training data (observation) $d \in D$. Normalizing r_{ij}, let

$$\omega_{ij} = \frac{r_{ij} s_j}{\sum_{k \in pa(i)} r_{ik} s_k} \tag{2.13}$$

While π_{ij} is the (normalized) prior probability that j is i's parent, ω_{ij} is the normalized posterior probability that j is i's parent. The learning rules are

$$\frac{\partial}{\partial \log c_{ij}} \left(-\sum_{d \in D} \mathcal{F}_1(Q_1^d, \theta) \right) = \sum_{d \in D} E_{Q_1^d} \left[\omega_{ij} - \pi_{ij} \right] \tag{2.14}$$

$$p_{ij}^{new} = \frac{\sum_{d \in D} E_{Q_1^d}[\omega_{ij} s_i]}{\sum_{d \in D} E_{Q_1^d}[\omega_{ij}]} \tag{2.15}$$

Updating c_{ij} and p_{ij} with the learning rules will also always increase $-\mathcal{F}_1$.

2.3.2 A More Complex Q

The naive mean field approximation (2.7) gave rise to an infinite expected energy of the system due to the fact that μ_{ij} was independent of s_j. Suppose we use a similar form for Q, but making μ_{ij} be dependent on s_j :

$$Q_2(\Lambda, \mathcal{H}) = \prod_{i \in I_H} {\sigma_i}^{s_i} (1 - \sigma_i)^{1 - s_i} \prod_{i \in I} \prod_{j \in pa(i)} \mu_{ij}(\mathcal{H})^{\lambda_{ij}}. \tag{2.16}$$

We require that $\mu_{ij}(\mathcal{H}) \geq 0$, $\sum_{j \in pa(i)} \mu_{ij}(\mathcal{H}) = 1$ and most importantly, $\mu_{ij}(\mathcal{H}) = 0$ if $s_j = 0$. One possibility for this is

$$\mu_{ij}(\mathcal{H}) = \frac{\nu_{ij} s_j}{\sum_{k \in pa(i)} \nu_{ik} s_k} \tag{2.17}$$

where ν_{ij} are new variational parameters which determine $\mu_{ij}(\mathcal{H})$ hence the distribution of λ_{ij} given the activations of the potential parents. Note the similarity in form of (2.17) to (2.2) and (2.13). We made $\mu_{ij}(\mathcal{H})$ be dependent only on the potential parents of i. This is consistent with the conditional dependencies of λ_{ij}.

The variational free energy is

$$\begin{aligned} -\mathcal{F}_2(Q_2, \theta) = E_{Q_2} \Big[\sum_{i \in I} \sum_{j \in pa(i)} \mu_{ij}(\mathcal{H}) \log \frac{c_{ij} s_j}{\sum_{j \in pa(i)} c_{ij} s_j} {p_{ij}}^{s_i} (1 - p_{ij})^{(1 - s_i)} \Big] - \\ \sum_{i \in I_H} \left(\sigma_i \log \sigma_i + (1 - \sigma_i) \log (1 - \sigma_i) \right) - \sum_{i \in I} \sum_{j \in pa(i)} E_{Q_2} \Big[\mu_{ij}(\mathcal{H}) \log \mu_{ij}(\mathcal{H}) \Big] \end{aligned} \tag{2.18}$$

Notice that now the expectations only range over s_i's, as Λ is again automatically integrated out.

Differentiating $\mathcal{F}_2$ with respect to μ_{ij} we obtain the mean field equations

$$\nu_{ij} \propto c_{ij} {p_{ij}}^{s_i} (1 - p_{ij})^{1-s_i}. \tag{2.19}$$

As with c_{ij}'s, the ν_{ij}'s are scale-invariant. If c_{ij} is the unnormalized prior probability that j is i's parent, ν_{ij} is the unnormalized posterior probability that j is i's parent, and is obtained by multiplying c_{ij} by the likelihood of generating s_i, that is, $p_{ij}^{s_i}(1-p_{ij})^{1-s_i}$. Further, μ_{ij} is the normalized posterior probability that j is i's parent.

To determine the inference rules, assume that ν_{ij} for $j \in pa(i)$ are always updated just before σ_i, so that when we update σ_i, $\nu_{ij} \propto c_{ij} {p_{ij}}^{s_i} (1-p_{ij})^{1-s_i}$. Differentiating (2.18) with respect to σ_i, we get

$$\sigma_i = sigmoid \left(\begin{array}{c} \sum_{j \in pa(i)} E_{Q_2}\Big[\mu_{ij}\Big] \log \frac{p_{ij}}{1-p_{ij}} + \\ \sum_{l \in ch(i)} E_{Q_2}\Big[\log \sum_{j \in pa(l)} \nu_{lj} s_j - \log \sum_{j \in pa(l)} c_{lj} s_j \Big]_{\sigma_i=0}^{\sigma_i=1} \end{array} \right) \tag{2.20}$$

The learning rules are

$$\frac{\partial}{\partial p_{ij}} \left(-\sum_{d \in D} \mathcal{F}_2(Q_2^d, \theta) \right) = \sum_{d \in D} E_{Q_2^d}\Big[\mu_{ij}\Big] \Big(\frac{\sigma_i}{p_{ij}} - \frac{1-\sigma_i}{1-p_{ij}} \Big) \tag{2.21}$$

$$p_{ij}^{new} = \frac{\sum_{d \in D} E_{Q_2^d}[\mu_{ij}]\sigma_i}{\sum_{d \in D} E_{Q_2^d}[\mu_{ij}]} \tag{2.22}$$

$$\frac{\partial}{\partial \log c_{ij}} \left(-\sum_{d \in D} \mathcal{F}_2(Q_2^d, \theta) \right) = \sum_{d \in D} E_{Q_2^d}[\mu_{ij} - \pi_{ij}] \tag{2.23}$$

where Q_2^d is the approximate posterior given observations $d \in D$.

For inference we need to compute the ν_{ij} and σ_i for $i \in I, j \in pa(i)$ such that Q_2 minimizes the free energy $\mathcal{F}_2$. This is achieved iteratively using the mean field equations (2.19) to update ν_{ij} and (2.20) to update σ_i. Since ν_{ij}'s determine μ_{ij}, which describes the distribution of Λ, we can view updating ν_{ij} using (2.19) as a segmentation step. Since σ_i determines the probability that a certain object corresponding to node i is present, update σ_i using (2.20) can be viewed as a recognition step. Since ν_{ij} and σ_i are updated one after another iteratively, the segmentation and recognition steps are intertwined. A similar argument holds for the mean field approximation of section 2.3.1. In that case, computing r_{ij} and ω_{ij} is the segmentation step, while updating σ_i using (2.12) is the recognition step.

2.3.3 Approximations

For an efficient implementation of credibility networks using variational approximations, we still need to evaluate terms of the form $E[\log x]$ and $E[1/x]$ where x is a weighted sum of binary random variates. The simplest approximation would be to bring the expectation inside the logarithm and inverse, i.e.,

$$E[\log x] \approx \log E[x] \qquad\qquad E[1/x] \approx 1/E[x] \tag{2.24}$$

In the case of the logarithm, the approximation always over-estimates by Jensen's Inequality. In the case of the inverse, the approximation always under-estimates due to the convexity of $1/x$. Although biased, these approximations work well enough in general. In fact the simulation results in chapter 3 were obtained using an implementation based on these approximations. However using these approximations, there is a bias towards the extreme values of 0 and 1 when solving for the means σ_i, since the energy terms in the mean field equations of σ_i will always be exaggerated.

We can obtain a better approximation by using the Taylor expansion of a function. Suppose we want to approximate $E[f(x)]$. Expanding $f(x)$ about the mean $x_o = E[x]$, we have

$$E[f(x)] = E[\sum_{k=0..\infty} \frac{f^{(k)}(x_o)}{k!}(x_o - x)^k] = \sum_{k=0..\infty} \frac{f^{(k)}(x_o)}{k!} E[(x_o - x)^k] \tag{2.25}$$

where $E[(x_o - x)^k]$ is the kth central moment of x about x_0. Using just the constant term in the Taylor expansion to approximate the expectation, we get $E[f(x)] \approx f(E[x])$, obtaining the simple approximations (2.24). A better approximation can be had if we use the first 3 terms of the Taylor expansion : $E[f(x)] \approx f(E[x]) + \frac{1}{2} f^{(2)}(E[x]) Var(x)$, where $Var(x)$ is the variance of x.

2.4 Relation to Some Models

2.4.1 Hierarchical Community of Experts

The hierarchical community of experts model [20, 35] is a generative model made up of pairs units consisting of a binary unit and a linear unit with Gaussian noise. Each

binary unit gates the output of the corresponding linear unit. One view of the model is that the binary units form a sigmoid belief network (SBN) which dynamically synthesizes a network of linear Gaussian units by gating out those linear units whose binary unit partners are off. This is similar to credibility networks where the parenthood units synthesize a tree structure for the presence units.

This view can aid in designing credibility networks which deal with instantiation parameters. We can consider a network of triples of units consisting of a parenthood unit i^*, a presence unit i, and an instantiation parameters unit i^+. The parenthood and presence units together form a binary credibility network which synthesizes the parse tree, which is a tree structured network of instantiation parameters units i^+. Then the values of instantiation parameters of the objects in the parse tree are generated from the instantiation parameters of the respective parents. Let the instantiation parameters at i^+ be denoted by the (multivariate) random variate γ_i and let $\Gamma = \{\gamma_i\}_{i \in I}$. The joint distribution is

$$\begin{aligned} P(\Lambda, S, \Gamma|\theta) &= P(\Lambda, S|\theta) P(\Gamma|\Lambda, S, \theta) \\ &= P(\Lambda, S|\theta) \prod_{i \in I} P(\gamma_i|\gamma_j \text{ where } \lambda_{ij} = 1) \end{aligned} \tag{2.26}$$

The $P(\Lambda, S|\theta)$ term is just the joint distribution of the binary credibility network (2.4). Each factor $P(\gamma_i|\gamma_j \text{ where } \lambda_{ij} = 1)$ in the product is parameterized by the weight α_{ij} from the parent node j^+ to i^+.

As an example, suppose the instantiation parameters are multivariate Gaussians. Suppose $\lambda_{ij} = 1$. Let γ_i have mean $f_{ij}(\gamma_i) = A_{ij}\gamma_j + b_{ij}$ and variance Σ_{ij} where A_{ij} is a matrix, b_{ij} is a vector and A_{ij}, b_{ij} and Σ_{ij} comprise α_{ij} :

$$P(\gamma_i|\gamma_j \text{ where } \lambda_{ij} = 1) = (2\pi)^{-\frac{n}{2}} |\Sigma_{ij}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\gamma_i - f_{ij}(\gamma_j))^T \Sigma_{ij}^{-1} (\gamma_i - f_{ij}(\gamma_j))\right) \tag{2.27}$$

Note that each connection between a parent and a unit in the parse tree is now a factor analyzer and the parse tree now consists of a hierarchy of factor analyzers. When inferring the parse tree, the analyzers at each layer compete for the control of the analyzers at the layer below. This is the hierarchical mixtures of factor analyzers we sought after in section 1.3.4.

2.4.2 Cooperation Versus Competition

One dimension along which we can differentiate and analyze models is the degree of cooperation or competition among the units of the network. Cooperative networks tend to produce representations of data that are distributed across the network, while competitive networks tend to produce sparse representations. For example, in factor analysis, all hidden units cooperate in influencing the activation of the visible units (by a linear combination of their activations). On the other hand, in mixtures of Gaussians, only one unit is responsible for the activations of the visible units, and there is competition among the hidden units for this responsibility. It is important to balance these two factors to produce representations that are both sparse and distributed [1, 18].

When designing networks of binary units, a design choice that has to be made is the combination function (called the mixing function in [39]). The combination function is analogous to the condensation rule of latent response models (LRMs) [27]. They describe how multiple factors combine in affecting the output of a response variable (or a unit below). The standard combination function is a weighted sum followed by a sigmoid squashing function :

$$x_i = sigmoid\Big(\sum_{j \in pa(i)} w_{ij} x_j\Big) \tag{2.28}$$

where x_i is the output of unit i and w_{ij} is the weight from j to i. This combination function permits errors made by some units to be corrected by other units without penalty. Hence there is little global pressure for each unit to produce a correct prediction. Hence such a combination function makes the units too cooperative and results in an obscured representation of the data.

Saund [39] proposed a new combination function, the noisy-OR that is more competitive :

$$x_i = 1 - \prod_{j \in pa(i)} (1 - c_{ij} x_j) \tag{2.29}$$

where x_i can be interpreted as the mean of an associated binary variable s_i and the weight c_{ij} can be interpreted as the probability that $s_i = 1$ given that $s_j = 1$, assuming

that the s_j's are independent. However, Dayan and Zemel [9] showed that the noisy-OR is still too cooperative because it allows multiple features to cooperate in causing the activation of a unit in a layer below. For example, if i has two parents, and both are on, and $c_{ij} = .5$ for both parents, then $x_i = .75$, although each parent individually predicted a value of $x_i = .5$. Now if the desired value of x_i is indeed .75, then both parents are not penalized even though both gave wrong predictions.

Dayan and Zemel's solution is for exactly one feature to cause the activation of a unit at a time (but multiple features could be affecting the activities of the layer below). For each unit, it first chooses one cause from among the potential causes (its parents). Then the activation of the unit is set according to which parent it has chosen. The combination function for a write white-black model (as in this thesis) they proposed is :

$$x_i = \frac{t_{*i} + \sum_{j \in pa(i)} b_{ij} x_j t_{ij}}{1 + \sum_{j \in pa(i)} b_{ij} x_j} \tag{2.30}$$

where b_{ij}'s describe the distribution from which the unit chooses its cause and t_{ij} is the activation of the unit if it chose cause j. It is easy to see that the b_{ij}'s are essentially credibilities and the t_{ij}'s are essentially the probability weights. If we assume that the credibility of the "grandparent" unit is 1 and its outgoing probability to unit i is t_{*i}, (2.30) is essentially (2.5). The difference is that while Dayan and Zemel's is a feedforward model, ours is a graphical model. While their model came about as a result of trying to increase competition among units, ours arised as a solution to the problems of segmentation and recognition.

This means that an advantage of our model is that there is more competition among the units than standard models. As Dayan and Zemel showed, this extra competition will produce internal representations that are more sparse, and yet still maintain a distributed nature, where multiple causes affect the activities of the layer below simultaneously.

Chapter 3

Experimental Data

3.1 The Bars Problem

The binary bars problem was proposed as a benchmark toy problem to test multilayer generative models, as well as to serve as a simple example of the necessity for multilayer models. The images are of size 6x6 and contain either horizontal bars or vertical bars, the chances for each orientation being the same. For each orientation, there are 6 possible bars each present independently half the time. Figure 3.1 shows a sample of images.

The optimal solution to the problem requires 2 hidden layers. The top layer contains one unit, which determines whether the bars are horizontal or vertical depending on whether it is on or off. Then the middle layer requires 12 units, 6 of which encode the 6 horizontal bars, the other 6 encoding the vertical bars. The fact that the vertical

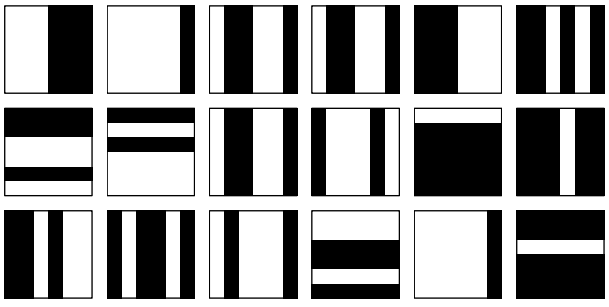


Figure 3.1: Sample images from the bars problem.

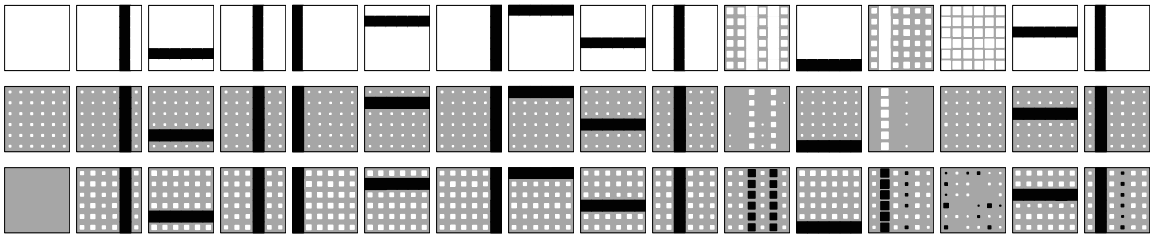


Figure 3.2: Hidden to output weights.

and horizontal bars are mutually exclusive means that a second layer of hidden units is required. The only way around this is to encode every possible combination of bars in the first layer, but this is not a more compact representation than storing the images themselves.

A 36-16-4 binary credibility network is trained on the set of all 128 images for 60 iterations. This takes around 50 seconds on an SGI R10000. After training, the network manages to use 12 of the middle layer units to encode each of the 12 bars and 1 top layer unit to encode the orientation. The unused units either are permanently off or have low credibilities and hence do not affect the rest of the units. The weights of the 16 middle layer units are shown in figure 3.2[1]. The top row shows $(p - .5)$ where p are the outgoing probabilities; the bottom row shows $\log c$ where c are the outgoing credibilities; and the middle row shows values of $c(p - .5)$. The middle row gives a nice summary of the feature encoded by each unit. Since the probabilities are mostly 0 or 1, the sign of each weight in the row describes whether the probability is 0 or 1, while the magnitude describes the credibility. Notice that the features are all localized. That is, each unit have high outgoing credibilities only for those pixels forming the corresponding bar.

The weights connecting the grandparent unit (1) and the used top layer units (v and h) to a horizontal bar unit (h') and a vertical bar unit (v') in the middle layer are shown in figure 3.3. Both 1 and v are permanently on, while h is on when the image contains

[1] To save ink, all Hinton diagrams in this thesis are printed in reverse video – black pixels means large values and white pixels means small values.

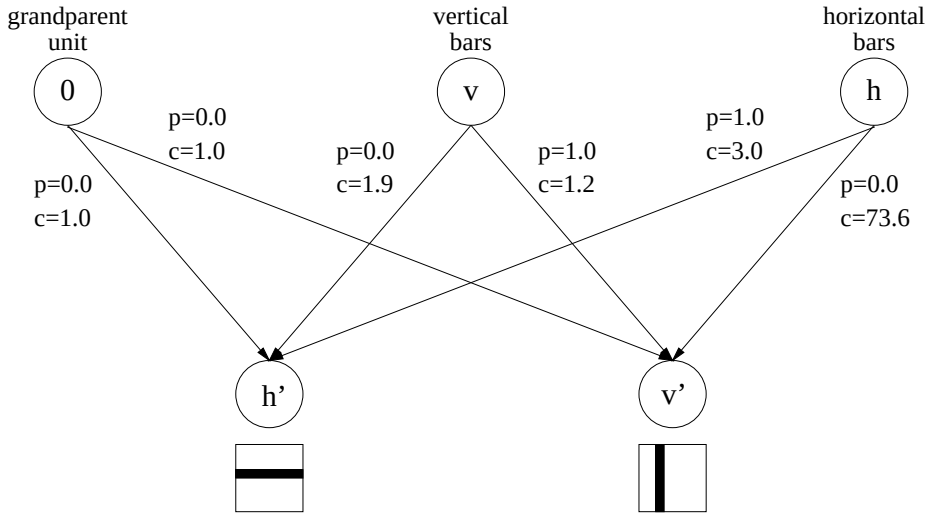


Figure 3.3: Encoding mutual exclusion between horizontal bars and vertical bars.

horizontal bars. Notice that when h is off, i.e. $s_h = 0$, the probability that h' is on is

$$\frac{c_{h'1}p_{h'1}s_1 + c_{h'v}p_{h'v}s_v}{c_{h'1}s_1 + c_{h'v}s_v} = \frac{0.0 + 0.0}{1.0 + 1.9} = 0.0$$

while the probability that v' is on is

$$\frac{c_{v'1}p_{v'1}s_1 + c_{v'v}p_{v'v}s_v}{c_{v'1}s_1 + c_{v'v}s_v} = \frac{0.0 + 1.2}{1.0 + 1.2} \approx 0.55 \approx 0.5$$

So when h is off all horizontal bars are turned off, while vertical bars are each independently present about half the time. Note that when v' is on its parent is v with probability 1, while when v' is off its parent is 1. Similarly when h is on, the horizontal bars are present about half the time, while the vertical bars are turned off due to the high credibilities and low probabilities h has for the vertical bar units.

Note that the probabilities p_{ij} from the top layer units to the middle layer units in figure 3.3 are either 0 or 1, and values in between (for example, 0.5 as in the bars problem) are obtained by mixing appropriate portions of the probabilities with c_{ij}'s serving as the mixing proportions. This shows that units in a credibility network are not totally competitive – the parents of a unit i can still cooperate to determine the activation of i. However, this cooperation is limited in the sense that the probability of i being on must always be a weighted sum of the incoming probabilities p_{ij} for $j \in pa(i)$.

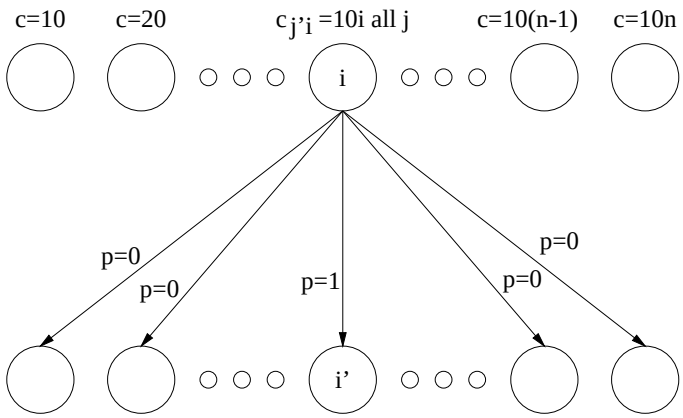


Figure 3.4: Encoding mutual exclusion between n units.

The above shows how the trained credibility network manages to encode the mutual exclusion between two features – the horizontal and vertical bars. This can be generalized to encode more than 2 mutually exclusive features. Figure 3.4 gives the general idea of how this can be done. There are n units in both the top and bottom layers. The top layer units are labeled $1, \ldots, n$, while the bottom layer units are labeled $1', \ldots, n'$. The outgoing credibilities are $c_{j'i} = 10i$ for all j' and the probabilities are $p_{j'i} = 1$ if $j' = i$ and $p_{j'i} = 0$ otherwise. If $s_n = 1$, because the outgoing credibilities $c_{j'n} = 10n$ are so high, it dominates the other units and the outgoing probabilities make sure that n' is on while the rest are off. If $s_n = 0$ and $s_{n-1} = 1$ then unit $n-1$ dominates the other units, and only $(n-1)'$ is on. For the network to work properly, the credibilities have to be very large. If we train a network instead of setting the weights manually, the network will not learn such exotic weight settings. This is because coding n mutually exclusive equiprobable features requires $\log_2 n$ bits while coding n independent features each with probability $\frac{1}{n}$ requires $\log_2 n + (n-1)\log_2 \frac{n}{n-1}$ bits – we need to code the fact that one of the features is present, while the other $n-1$ are absent. Bounding $\log x \leq x - 1$, the number of extra bits required is

$$(n-1)\log_2 \frac{n}{n-1} \leq \frac{1}{\log 2}(n-1)\left(\frac{n}{n-1} - 1\right) = \frac{1}{\log 2}$$

So at most $1/\log 2 \approx 1.44$ extra bits are required. For large n this is negligible.

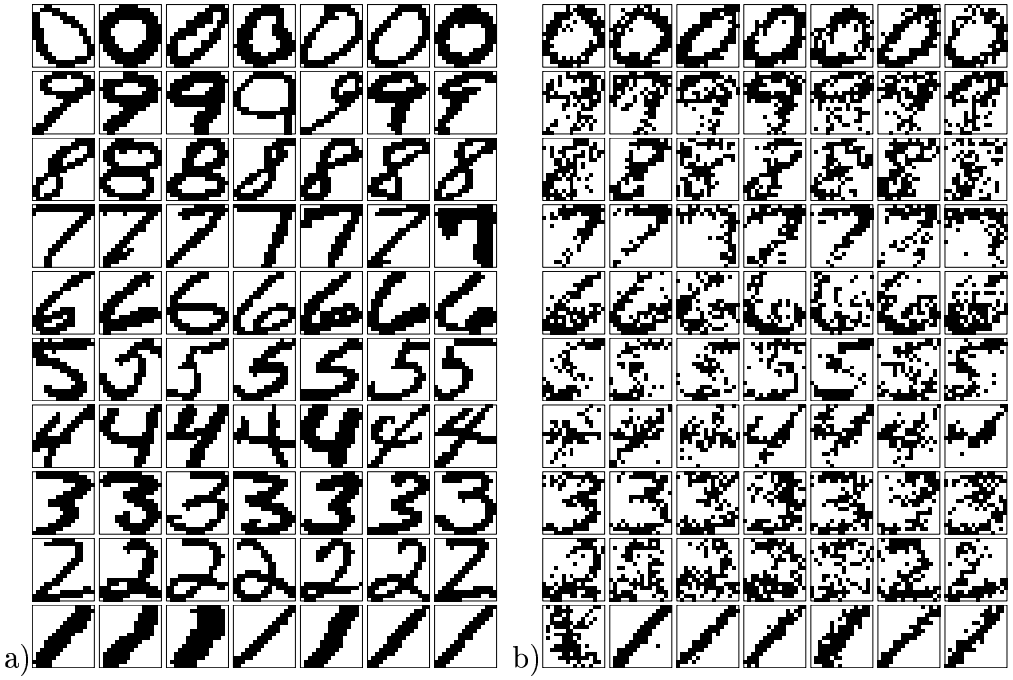


Figure 3.5: a) Sample images of digits used to train and test the ML classifier. b) Sample images generated by the trained models.

3.2 Classifying Hand-written Digits

We also tested credibility networks on a more realistic problem. We constructed a maximum likelihood classifier for 16 by 16 binary hand-written images. The data was obtained from the CEDAR CDROM 1 database [22], and was preprocessed to give a set of 11000 16 by 16 binary images by Michael Revow. There are 1100 images of single digits in each of ten [0-9] digit classes. The pixels forming the digits have value 1 while the pixels for the background have value 0. Example images are shown in figure 3.5a). For each digit class, 700 randomly chosen images were used to train a 256-24-8 network until convergence, taking 50 iterations. Batch learning with a learning rate of 0.1 for the credibilities was used. MAP training is employed, with the priors on the probability weights being Beta-$(1+e^{-3}, 1+e^{-3})$ distributions[2], while the log credibilities have a decay rate of 0.01. A grandparent unit with outgoing probabilities of 0 was used. This is both to encourage

[2] A Beta-(a, b) distribution is a distribution over $[0, 1]$ where the probability of $x \in [0, 1]$ is proportional to $x^{a-1}(1-x)^{b-1}$. Both a, b has to be positive.

	1	2	3	4	5	6	7	8	9	0
1	396	1	0	1	1	0	0	1	0	0
2	2	383	1	3	0	3	2	5	1	0
3	0	5	378	0	8	0	1	6	2	0
4	0	0	0	390	0	1	0	3	6	0
5	1	0	19	0	365	2	0	7	1	5
6	5	1	0	4	5	384	0	0	0	1
7	2	0	2	3	0	0	378	4	11	0
8	5	1	12	3	6	0	0	363	8	2
9	0	2	0	5	1	0	11	5	376	0
0	3	1	0	1	0	1	0	2	0	392

Table 3.1: Confusion matrix of ML classifier. The ij^{th} entry is the number of times an image of digit i was classified as digit j.

sparse coding and to code the value of the pixels in the background of the images. Sample images generated from the trained models are shown in figure 3.5b).

After training, the 400 unused images from each digit class was used for testing. Each test image was classified using the model which assigned the lowest variational free energy to it. Table 3.1 shows the confusion matrix. The error rate is 4.9%. As comparison, the error rate of the 1-nearest neighbor algorithm on the same training and testing data has an error rate of 5.3%. This shows that on this task credibility networks are at least competitive with other models, although it is clearly not excellent (compared with Saul and Jordan's work with variational methods for SBNs [37, 38] and from the poor quality of generated images in figure 3.5b)).

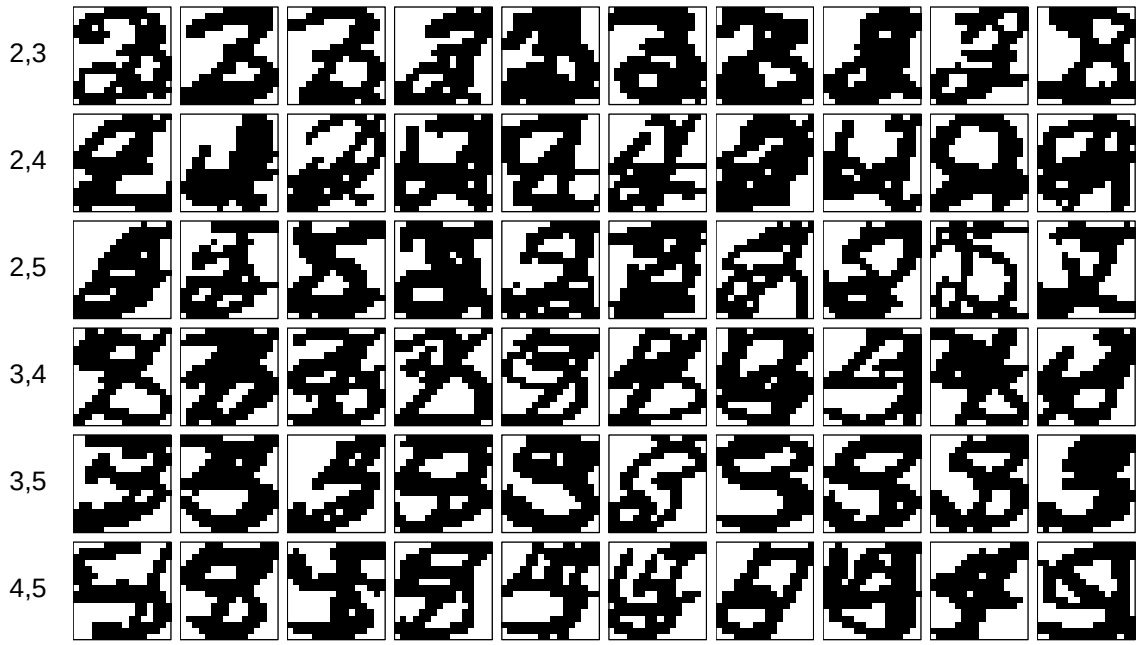


Figure 3.6: Sample images from the test set. The classes of digits of the images in a row are given to the left.

3.3 Segmenting Hand-written Digits

Hinton and Revow [19] used a mixture of factor analyzers model to segment and estimate the pose of digit strings. Images of size 7x16 containing 3 digits from the classes 2,3,4 and 5 are used. Each digit is roughly of size 5x5. When the digits do not overlap, the model was able to identify the digits present and segment the image easily. The hard cases are those in which two or more digits overlap significantly. To assess the ability of credibility networks at segmenting hand-written digits, we used superpositions (ORs) of digits at exactly the same location. This problem is much harder than segmenting digit strings which may or may not overlap, but also easier to implement. Example images of the superposition of each combination of two digits are given in figure 3.6.

We trained a single credibility network with images from digit classes 2,3,4 and 5. Then images of one or two digits from distinct classes were shown to the network for classification.

The data used is the set of 16 by 16 images of single digits from the previous section.

a)

	2	3	4	5
2	392	3	3	2
3	12	372	1	15
4	5	0	394	1
5	3	40	3	354

b)

	3	4	5
2	0.95	0.80	0.85
3		0.65	0.95
4			0.50

c)

	3	4	5
2	0.80	0.90	0.70
3		0.80	0.95
4			0.70

Table 3.2: a) Confusion matrix for single digit classification. The ij^{th} entry is the number of times an image of digit i was classified as digit j. b) Percentage of cases inferred correctly by the trained network for each combination of two digits. c) Similarly for the author.

The training set consists of 700 images of single digits for each digit class 2, 3, 4 and 5. The size of the credibility network is 256-64-4. The 64 middle layer units are meant to encode low level features, while each of the 4 top level units are meant to encode a digit class. Without supervision, the network was not able to discriminate among the different digit classes in the top level units. To aid the network, we clamped at 1 the activation of the top layer unit corresponding to the class of the digit in the current image while fixing the rest at 0 when showing the network each image for training.

After training, the network is tested on images of single digits not in the training set and it achieved an error rate of 5.5%. The predicted class of each image was taken to be the class corresponding to the top layer unit with the highest activation. The confusion matrix is given in table 3.2a). To test the ability to segment, we showed the network images of 2 overlapping digits from distinct classes. The predicted classes of the two digits are chosen to be the corresponding classes of the 2 top layer units with the highest activation. The test set consists of 120 images, 20 images per combination of two classes. Figure 3.6 gives a number of tests. Table 3.2b) gives the percentage of tests inferred correctly for each two digit class combination. A human subject (namely the author) was tested on the same test set and his performance is shown in table 3.2c). The network achieved an error rate of 21.7% while the author erred on 19.2% of the images.

We can in fact obtain more than just the classes of digits present in each image. We can produce a segmentation of the image into an image for each class present. Recall that

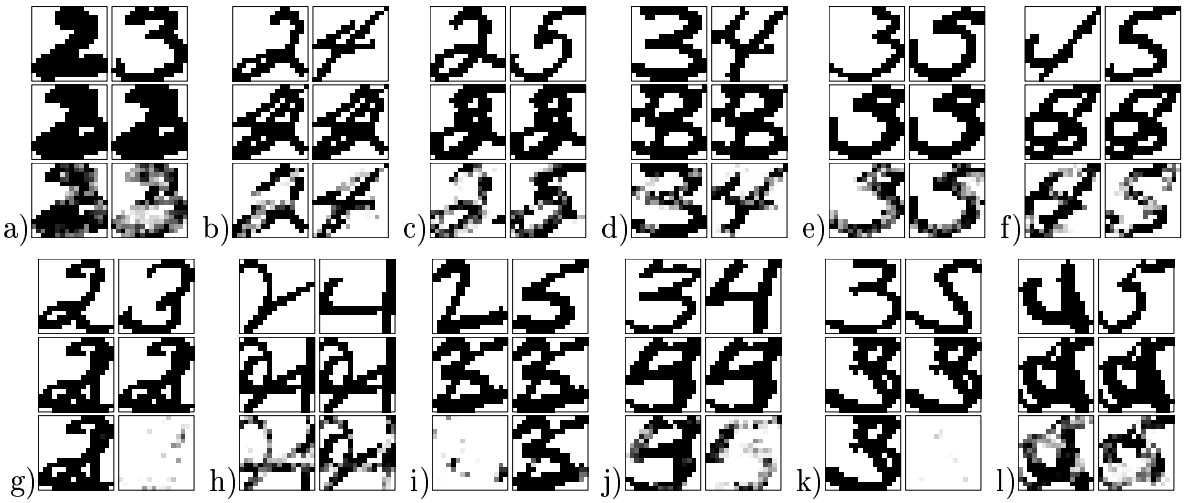


Figure 3.7: Segmentations of pairs of digits. In each group, the top two images are the original images. The middle two images are both superpositions of the top two images. The bottom two images are the segmented images produced by the credibility network.

given the values of S the posterior probability of unit j being unit i's parent is μ_{ij}. Then the posterior probability of pixel i belonging to digit class k is proportional to $\sum_j \mu_{ij}\mu_{jk}$. Integrating over the approximate posterior distribution Q, the probability that pixel i belongs to digit class k is $\sum_j E_Q[\mu_{ij}\mu_{jk}]$. We can approximate this using $E[1/x] \approx 1/E[x]$, so that the probability above is approximately proportional to $\sum_j \nu_{ij}\sigma_j\nu_{jk}\sigma_k$. This gives a simple way to determine how much each hidden unit contributes to the image. Figure 3.7 shows a number of segmentations. Note that for each pixel, the sum of the probabilities of the pixel belonging to each digit class is 1. To make the picture clearer, a white pixel means a probability of $\leq .1$ of belonging to a class, while black means $\geq .6$ probability, and the intensity of a grey pixel describes the size of the probability if it is between .1 and .6. Figures 3.7a) to 3.7f) shows successful segmentations, while figures 3.7g) to 3.7l) shows unsuccessful segmentations. On further inspection, it turns out that the cases of unsuccessful segmentations are those where the top level unit activations are uncertain, or those where only one top level unit has any activity. Notice in figure 3.7k) that the model believes only digit class 3 is present, but it believes there are two instances of the digit 3.

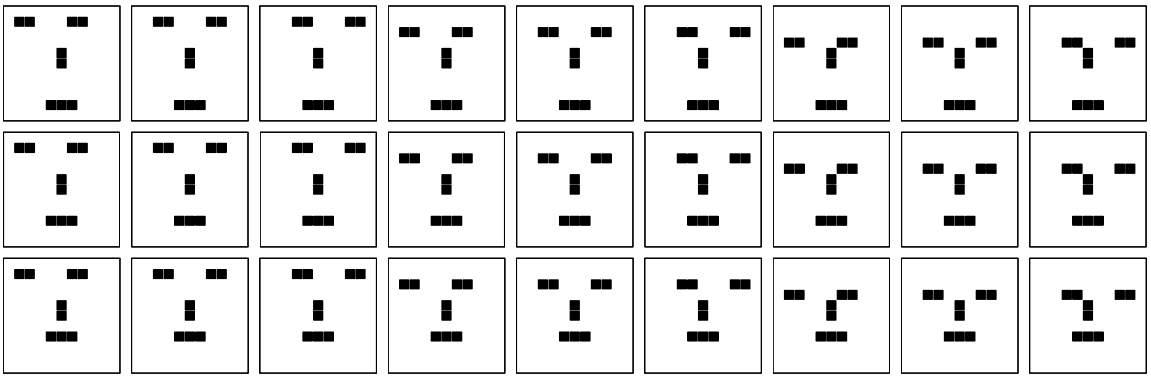


Figure 3.8: Some images from the faces problem.

3.4 The Faces Problem

We tested a three layer credibility network on a toy problem in which there was clear hierarchical structure. The image set consists of 243 11x11 binary bitmap images in which pixels are arranged in such a manner as to resemble faces. Figure 3.8 shows a subset of the images.

Each image shows a face. Each face consists of 3 features – a pair of eyes, a nose and a mouth. The x position of the eyes relative to the nose is perturbed randomly either to the left by one pixel, to the right by one pixel or unchanged, each with probability 1/3. Similarly, the y position of the eyes and mouth are perturbed up, down or unchanged equiprobably. Finally, the whole face is translated left, right or unchanged and up, down or unchanged. This creates 9 possible face locations given by the nose positions, each with 27 possible faces, all equiprobably. The images in figure 3.8 are the 27 possible faces with the noses at the center.

The faces problem is quite hard for credibility networks. There are many position-specific features and complex interactions between the features. There are 25 eye features, 9 nose features and 15 mouth features, a total of 49 low level position dependent features. Conditioned on a nose position, the eyes and mouth positions are independent. This allows a simple way to code the interactions if we have one high level unit corresponding to each nose position. However, conditioned on the eye positions, the nose and mouth positions are not independent. Given only the images, it is not trivial for top level units

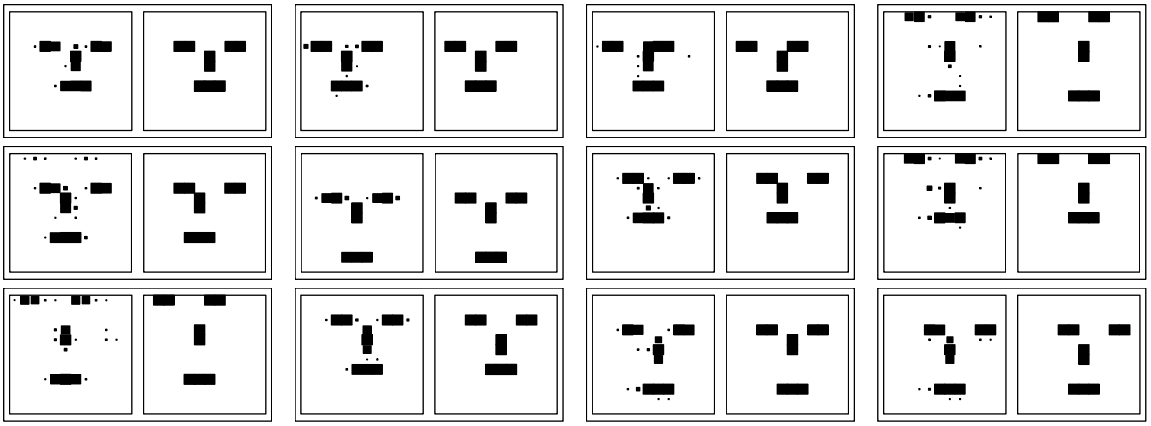


Figure 3.9: Image reconstructions of SBN. The left image in each group is the prediction made by the SBN, while the right image is the actual training image. The area of each pixel is the probability that the pixel is on.

to discover that conditioning on the nose positions gives the most efficient coding.

We first tried training a SBN on the faces image set. The network structure used is 121-50-10. We trained the network using Gibbs sampling for 10,000 passes through the training set in batch mode. This took over 24 hours of computation time. After training, the network used approximately 9000 bits to encode the images, with approximately 8000 bits used to encode the hidden unit activations. Of course these coding costs did not take into account the "bits-back" recovered through determining the random bits that led to the particular choice of hidden unit activations from the posterior distribution [21]. Determining the optimal amount of bits-back requires knowing the exact posterior distribution of the SBN, which is too expensive in our case. Instead we used a mean field approximation to the posterior distribution. The coding cost for the images using the mean field approximation is approximately 3500 bits with approximately 2950 bits allocated to coding the hidden unit activations. Only around 550 bits, or 2.3 bits per image, are used to code the images given the hidden unit activations, This says that the network was able to learn most low level features accurately. A direct way to verify this is to see the reconstructions of images made by the network given the activities of the hidden units. This is given in figure 3.9. The reconstructions match the actual images

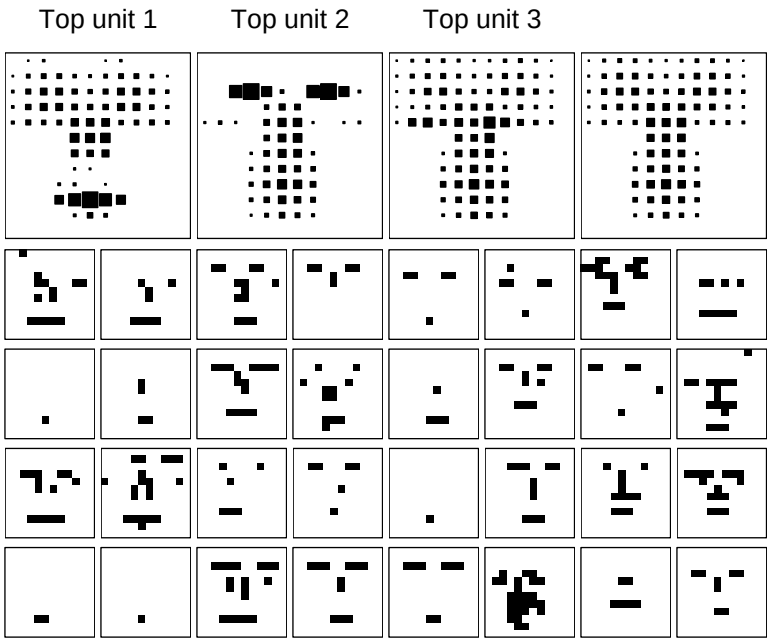


Figure 3.10: Images generated by SBN trained on the faces problem. The large diagrams are averages over the randomly generated images, with a large black pixel meaning that the pixel is on for almost all of the images generated. The eight small diagrams below each large diagram are samples from the randomly generated images.

quite accurately. However since the optimal coding cost is around 2000 bits[3], the network failed to encode the distribution over features efficiently. Analyzing the weights from the top layer units to the second layer units, we see that there are only three top layer units with at least one outgoing weight with magnitude greater than 1. Further, only the weights of these three units have a mean magnitude of at least 0.1. This means that the other top layer units have essentially been "turned off" and do not contribute to coding the images.

One way to have an idea of what redundancies have been captured by each active top layer unit is to look at images generated from the trained model. This is given in figure 3.10. As a control, we generated images from the model without any restrictions on the top layer units. This is shown in the rightmost set of diagrams in figure 3.10. First we

[3] $243 \log_2 243 \approx 1925.7$.

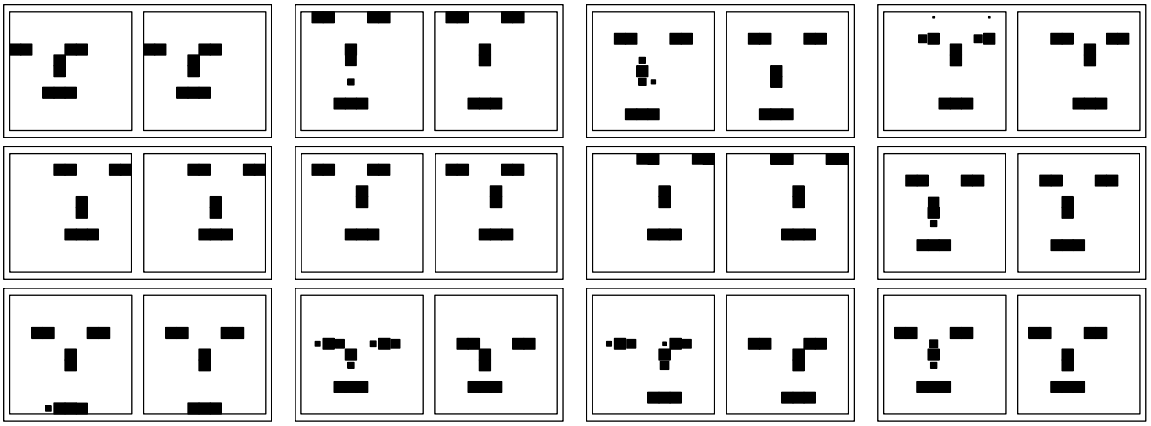


Figure 3.11: Predictions of the credibility network (left) and the actual images (right).

look at images generated when the activity of one of the active top layer units is fixed at 1 (the activities of other hidden units are sampled from the posterior given the activity of that top layer unit). This is shown in the leftmost set of diagrams in figure 3.10. Note that the mouth features all have the same y-coordinate. This shows up prominently in the averaged image, where the pixels are much larger than the corresponding pixels in the averaged image of the control. Further, mouth features in other locations are inhibited by the top layer unit and are almost absent from the images. The pixels in the averaged image corresponding to the mouth features at other locations are much smaller than the corresponding pixels in the control. The same effect is observed when we fix the activity of the second top layer unit at 1. This time, the top layer unit coded an eye location instead, and eye features at other locations are inhibited. The effects of the third top layer unit is not as obvious as the first two. From the averaged images, we see that eye features at the bottom left corner are slightly more likely to occur when the third top layer unit is on than when it is off. Note that the generated images have only a slight resemblance to the actual training data. One reason for this is that the features are not mutually exclusive.

We trained a similarly structured 121-50-10 credibility network on the image set using a mean field approximation. To encourage sparse coding we initialized the probability weights and mean field activations in the hidden layers at random values in the range

$[0, .2]$. The logarithms of the credibilities are randomly chosen from a zero mean unit variance Gaussian distribution. The learning rate for the log credibilities between the input layer and the first hidden layer is $20/D$ where D is 243, the number of training images. The learning rate for the weights between the first and second hidden layers is $0.1/D$. The learning rate for the weights for the second hidden layer does not matter, since these units have only one parent. No weight decay was used for the log credibilities, and a uniform prior for the probability weights was used. The network was trained for 2000 iterations. This took approximately 10 hours. After training, the network took approximately 3300 bits to code the images, with 300 bits used in reconstructing the images at the visible nodes. Note that the smaller coding cost does not mean that the credibility network is better than the SBN at encoding the images, because the computation of the coding costs involve approximations. The reconstructions of the credibility network on a a few images are shown in figure 3.11. The reconstructions of the credibility network are quite accurate, agreeing with the low coding costs for the image pixels.

To see what redundancies between features the top layer units have coded, we again analyze the images generated by the model. Five top layer units are either permanently on or off, so they do not code anything. The average activations of the other five units are approximately $0.42, 0.93, 0.73, 0.16$ and 0.70. Since the first and fourth of these units are off more often than on, we looked at images when these two units are on. The other units are more often on than off, so images generated when they are off will be more informative. The images are given in figure 3.12. Unit 4 codes faces with eyes in a particular location, although the effect is less pronounced than the effects of the top layer units in the SBN (figure 3.10). Unit 8 codes faces where the mouth and nose are in a particular location. Again the effect is less obvious. The other top layer units have weaker effects.

For both SBN trained using Gibbs sampling and credibility network trained using mean field approximation, the coding cost for the images is around 3300 bits, while the optimal coding cost is only around 2000 bits. One explanation for this is the lack of mutual exclusion between the units encoding features, which can be seen from the images

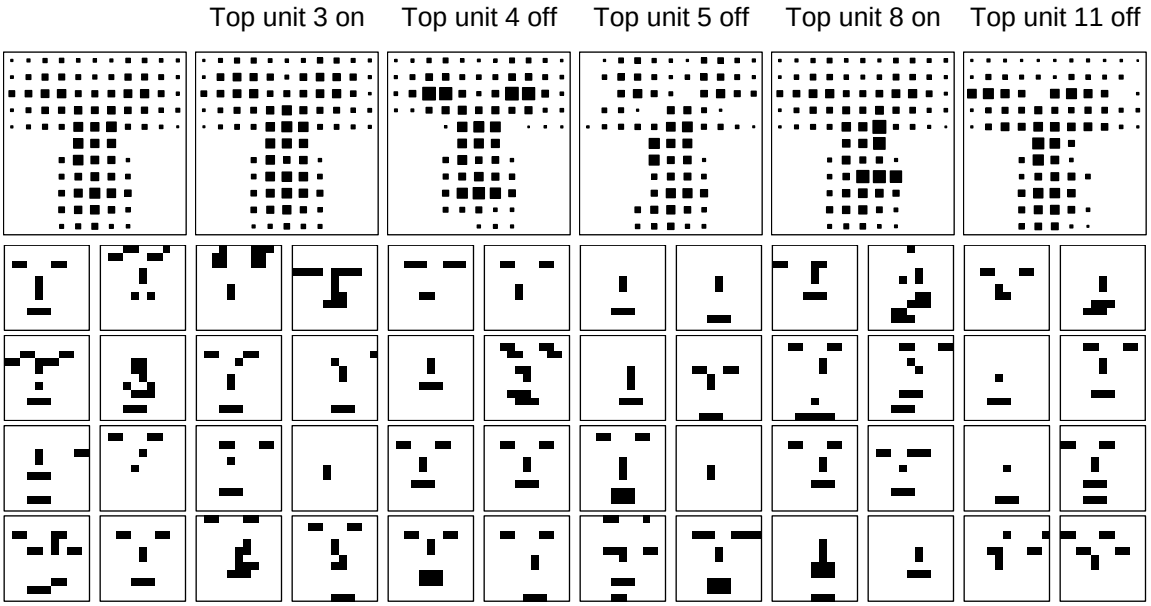


Figure 3.12: Images generated by credibility network trained on the faces problem. The large diagrams are averages over the randomly generated images, with a large black pixel meaning that the pixel is on for almost all of the images generated. The eight small diagrams below each large diagram are samples from the randomly generated images.

generated in figures 3.10 and 3.12. Consider the following generative model : faces in each of the 9 locations are present independently $\frac{1}{9}$ of the time. If a face is present, the corresponding nose is present, and eyes in each of the 9 locations are present again independently $\frac{1}{9}$ of the time. Similarly mouths in each of the 3 locations are present independently a third of the time. This model can be said to have captured the structure of the faces problem, except for the mutual exclusion among the features. If credibility network and SBN used in this section cannot capture mutual exclusion among features, this will be the best they could do. Now recall that coding one of n mutually exclusive equiprobable features under a model which assumes the features are independent requires $\log_2 n + (n-1)\log_2 \frac{n}{n-1}$ bits. So under this model the coding cost for each image is

$$\underbrace{\log_2 9 + 8\log_2 \frac{9}{8}}_{\text{face locations}} + \underbrace{\log_2 9 + 8\log_2 \frac{9}{8}}_{\text{eyes}} + \underbrace{\log_2 3 + 2\log_2 \frac{3}{2}}_{\text{mouths}} \approx 11.81 \text{ bits}$$

and the total coding cost for all the images is $243 \times 11.81 \approx 2871$ bits. This is only

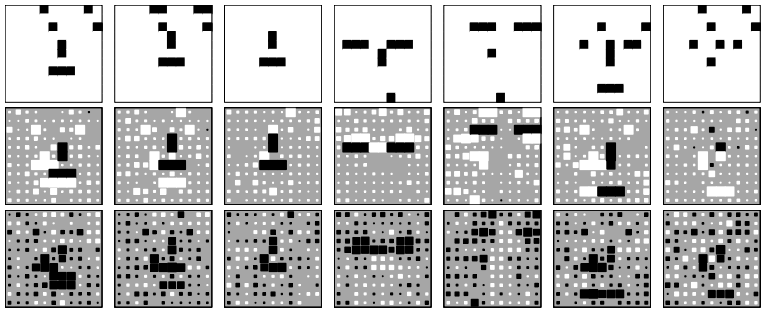


Figure 3.13: Hidden to output weights for some low level units.

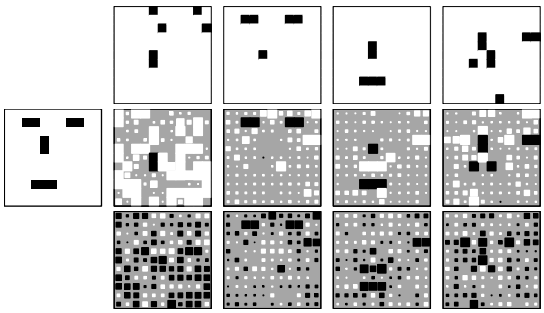


Figure 3.14: An example of how low level units interact in a complicated way.

roughly 450 bits less than the 3300 bits required by the trained credibility network to code the images, i.e. the credibility network used less than 2 bits more per image than the optimal.

The trained credibility network was actually able to deal with some of the mutual exclusion between the low level features. This is achieved by low level units which not only assert that certain pixels are on, they also assert strongly that certain other pixels are off. The outgoing weights of some example low level units are given in figure 3.13. The top row shows the probabilities p, the bottom row shows the logarithm of the credibilities $\log c$ and the middle row shows $c(p - .5)$. When these low level units are active, not only are the corresponding features present, because they assert that some neighboring pixels are off, features corresponding to those pixels are forced to be absent. There is a downside in dealing with mutual exclusion this way. When one feature is present, many units could be activated. The effects of many of these active units cancel each other out

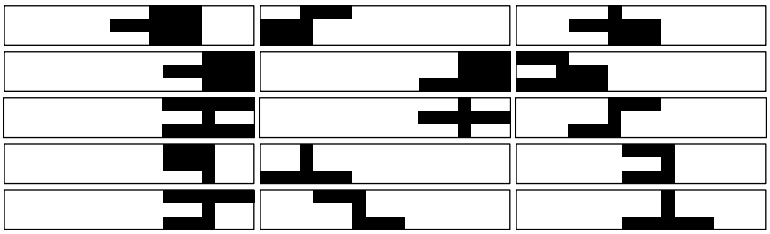


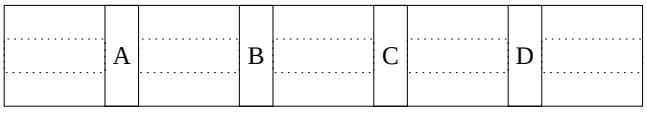
Figure 3.15: Sample images.

leaving only the feature that is present. This makes the response of the low level units given an image very complicated, and it makes it harder for the higher level units to capture the redundancies among the activations of the low level units. An example to show this is given in figure 3.14. The image to the left is the training image. The right shows the outgoing weights of the four low level units contributing to the reconstruction of the image. Note that the nose feature encoded by the first low level unit to the left is absent from the image. The effect of this unit is canceled out by the third and fourth units, while extraneous effects of the fourth unit are canceled by the second and fourth units.

3.5 Correcting The Faces Problem

The last experiment is designed as a simplification of the faces problem in section 3.4. In that section, two reasons why credibility networks failed to model the faces problem was given : the mutual exclusion among low level features, and the complicated effects of the low level units, which were caused by the network trying to capture some mutual exclusion among the low level features. This experiment shows that once these two reasons are removed credibility networks are able to perform well.

The images are 3 by 19 in size, and is made up of 9 regions as shown below.



First one of regions A, B, C and D is chosen and turned on. Then each region to the

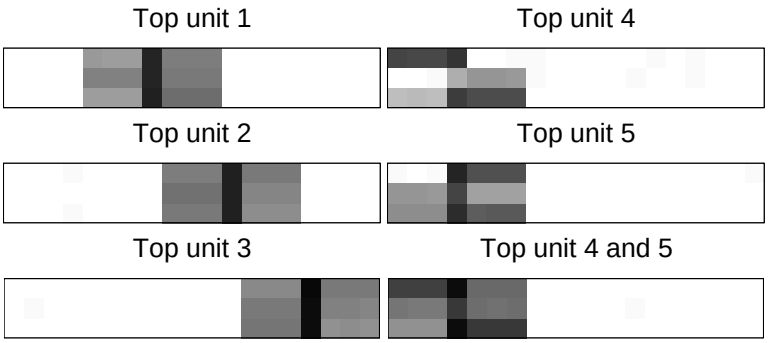


Figure 3.16: Each image is the average of 1000 images generated from the model when only the top level units given above each image are on.

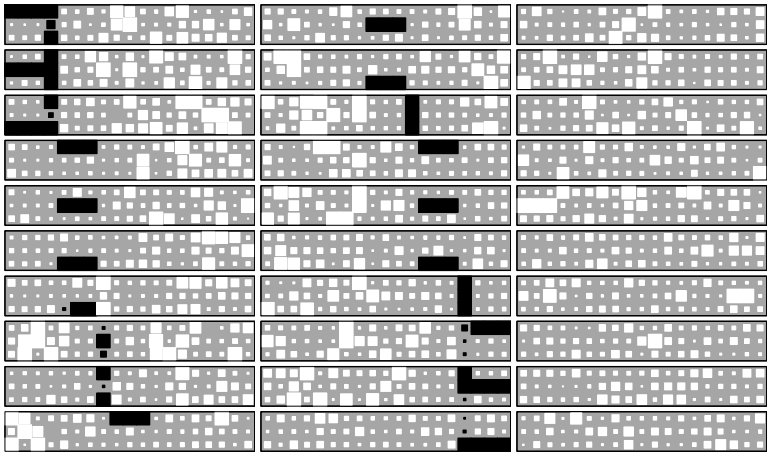


Figure 3.17: Hidden to output weights for all low level units.

left and right of the chosen region is split into 3 subregions as in the figure above. Each subregion is independently turned on half the time. Sample images are shown in figure 3.15. There are 256 possible images. Note that there still is mutual exclusion among the four groups of "bars" centered around regions A, B, C and D.

We trained a 39x30x6 network on the 256 images for 200 iterations. This took 13 minutes. After training, the network manages to correctly learn the generation process of the images except for the mutual exclusion among the four groups. Three high level units encoded three of the groups, one high level unit was unused and two were used to encode one of the groups. To see this we again analyze the images generated from the

model. The averages of the generated images are shown in figure 3.16. Note that top level unit 4 is used when the subregion in the top left corner is on, while top level unit 5 is used when the subregion in the top left corner is off. The low level units were used to encode the low level features, namely the bars. Figure 3.17 shows $c(p - .5)$ where c are the credibilities from the hidden units to the visible units and p are the probabilities. The units have been reordered for easier viewing. The coding cost was approximately 2540 bits, while the optimal coding cost was roughly 2370 bits (assuming a model which cannot handle the mutual exclusion among the four groups). The model used less than 1 bit more than the optimal to encode each image.

Chapter 4

Conclusions

Segmentation and recognition are important subtasks of image interpretation. The general approach to statistical image interpretation tries to solve each subtask separately. This introduces various problems as the subtasks are interrelated.

These problems can be avoided by solving both segmentation and recognition simultaneously. This can be achieved by viewing segmentation and recognition as subtasks of finding the correct parse tree of the image, and inferring the parse tree directly.

The credibility network is an instantiation of this idea. It is a graphical model which describes a probability distribution over all possible parse trees with the leaves forming the image pixels.

The parameters of a credibility network can be learned and inference can be achieved using variational approximations. During inference, the results of segmentation and recognition are iteratively improved.

4.1 Discussion

Using parse trees as the internal representations of images, credibility networks avoid the usual problems associated with a bottom-up approach to image interpretation.

Segmentation can be carried out in a statistically sound manner, removing the need for hand crafted ad hoc methods. Further, as segmentation and recognition results are improved iteratively, partial information about the objects present in the image can aid

in improving the current segmentation. Of course the other way of partial segmentation results improving the current recognition holds true too. This is demonstrated in the experiment of segmenting hand-written digits.

The granularity problem for segmentation is also resolved using parse trees. The parse trees describe the segmentations of the image at every level of granularity, from individual pixels to the whole image.

Credibility networks also have an advantage from a representational standpoint. In credibility networks, groups of units cooperate to form an image while competing for responsibility over each pixel. The cooperative nature of the units over the image means that the representations used are distributed in nature. While the competition over individual units below means that the representations are also sparse.

The feature encoded by a unit is localized to those units below which the unit has high credibility for. If two units are from different locales, the features represented are orthogonal to each other. However, if two units share the same locale, they compete for the locale.

Probabilistic features are possible. Consider a feature which requires a unit to be on with a probability of .5. In SBN, this is encoded by the active parents having weights which sum to 0. This also means that the parents are uncertain of the prediction of .5. If any parent unit misbehaves, then the probability of the unit deviates from .5. This means that the feature becomes non-localized – all parents are responsible for the unit. On the other hand, credibility networks can easily encode the credibility of predictions independently of the predictions themselves.

Credibility networks also have disadvantages. First of all, mean field learning and inference rules cannot be exactly computed, and we have to resort to approximations. Another problem is that the choice of parents of each unit is independent from the choice of parents of other units. This is normally not the case in images – for example, pixels near each other have a higher probability of having the same parent object than if they are far apart. This fact is not built into the system. This independence of parenthood choices also means that credibility networks cannot encode features that are mutually exclusive easily. As the faces problem shows, this is a very serious problem with credibility

networks.

At the meta level, while we think of parse trees from the top down (a scene is decomposed into objects, each object into its parts, each part into subparts), credibility networks construct parse trees from the bottom up, since it is the children which choose the parents. Since the children do not know their parents before they actually choose them, this shows up as mutual exclusion problems.

4.2 Future Work

One possible line of future research is to extend the current model in various directions. One direction is to incorporate instantiation parameters into the model as described in section 2.4.1. Many quantities of interest in image interpretation are continuous, for example, intensities, positions and orientations. The ability to extract these quantities from images is important in its own right. Further, encoding them can also help reduce the representational burden of the network by reducing redundancies. This is true even for binary images. For example, images of digits are translation-invariant. However, to encode a set of digits at various locations would require reproducing the same network structure over every location. If we can encode the position of the digits separately, then only one copy of the network is required.

Another possible extension would be in the time domain. The current model assumes that the images are identically and independently drawn from the same distribution. Given a single image, the network has to figure out the entire parse tree of the image. Consider instead using a sequence of images that change slowly over time. When a new image is presented to the network, it only has to figure out what changes occurred to the parse tree from the changes in the image. This is usually a much easier task because a sequence of images provides a lot more data and redundancies for the model to use.

But perhaps a more important line of research would be in addressing the shortcomings of credibility networks.

One problem is the approximations used in making the mean field learning and inference rules in section 2.3 tractable. These are quite gross approximations, and they are

not guaranteed to increase a lower bound on the log likelihood. Better approximations are needed. Better approximations described in section 2.3.3 can be implemented and compared with the current approximations. Markov Chain Monte Carlo methods like Gibbs sampling, which do not need any approximations, should be investigated.

Another problem is the independent parenthood choices of the units. This manifests itself as an inability to encode mutually exclusive features. A simple way to handle this might be to use lateral connections among units to force the mutual exclusion. A more useful solution is to have a more flexible distribution over parse trees while not making learning and inference even more intractable.

Bibliography

[1] H. Barlow. Unsupervised learning. *Neural Computation*, 1:295–311, 1989.

[2] C. A. Bouman and M. Shapiro. A multiscale random field model for Bayesian image segmentation. *IEEE Transactions on Image Processing*, 3(2):162–177, 1994.

[3] W. L. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.

[4] K. C. Chou, A. S. Willsky, and A. Benveniste. Multiscale recursive estimation, data fusion, and regularization. *IEEE Transactions on Automatic Control*, 39(3):464–478, 1994.

[5] G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks (research note). *Artificial Intelligence*, 42:393–405, 1990.

[6] T. Cover and J. Thomas. *Elements of Information Theory.* John Wiley, New York, 1991.

[7] Y. Le Cun, B. Boser, J. S. Denker, S. Solla, R. E. Howard, and L. D. Jackel. Back-propagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[8] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel. Helmholtz machines. *Neural Computation*, 7:1022–1037, 1995.

[9] P. Dayan and R. S. Zemel. Competition and multiple cause models. *Neural Computation*, 7(3):565–579, 1995.

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

[11] B. J. Frey, G. E. Hinton, and P. Dayan. Does the wake-sleep algorithm produce good density estimators? In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8. The MIT Press, 1995.

[12] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[13] Z. Ghahramani and G. E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-91-1, University of Toronto, Department of Computer Science, ftp://ftp.cs.utoronto.ca/zoubin/mfa.tar.gz, 1996.

[14] T. J. Hastie, P. Y. Simard, and E. Sackinger. Learning prototype models for tangent distance. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7. The MIT Press, 1994.

[15] D. Heckerman. A tutorial on learning with Bayesian networks. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer Academic Publishers, 1998.

[16] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The wake-sleep algorithm for self-organizing neural networks. *Science*, 268:1158–1161, 1995.

[17] G. E. Hinton, P. Dayan, and M. Revow. Modeling the manifolds of images of handwritten digits. *IEEE Transactions on Neural Networks*, 8:65–74, 1997.

[18] G. E. Hinton and Z. Ghahramani. Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society of London B*, 352:1177–1190, 1997.

[19] G. E. Hinton and M. Revow. Using mixtures of factor analyzers for segmentation and pose estimation, 1997.

[20] G. E. Hinton, B. Sallans, and Z. Ghahramani. A hierarchical community of experts. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer Academic Publishers, 1998.

[21] G. E. Hinton and R. S. Zemel. Autoencoders, minimum description length and helmholtz free energy. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan Kaufmann Publishers, San Mateo CA, 1993.

[22] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994.

[23] W. W. Irving, P. W. Fieguth, and A. S. Willsky. A overlapping tree approach to multiscale stochastic modeling and estimation. *IEEE Transactions on Image Processing*, 1995.

[24] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer Academic Publishers, 1998.

[25] M. R. Luettgen and A. S. Willsky. Likelihood calculation for a class of multiscale stochastic models, with application to texture discrimination. *IEEE Transactions on Image Processing*, 4(2):194–207, 1995.

[26] D. J. C. MacKay. Introduction to monte carlo methods. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer Academic Publishers, 1998.

[27] E. Maris, P. De Boeck, and I. Van Mechelen. Probability matrix decomposition models. *Psychometrika*, 61(1):7–29, 1996.

[28] D. Marr. *Vision : A Computational Investigation into the Human Representation and Processing of Visual Information*. W. H. Freeman and company, San Francisco, 1980.

[29] R. M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113, 1992.

[30] R. M. Neal. Probabilistic inference using markov chain monte carlo methods. Technical report, University of Toronto, Department of Computer Science, 1993.

[31] R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental and other algorithms. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer Academic Publishers, 1997.

[32] G. Parisi. *Statistical Field Theory*. Addison-Wesley, Redwood City, 1988.

[33] J. Pearl. *Probabilistic reasoning in intelligent systems : networks of plausible inference*. Morgan Kaufmann Publishers, San Mateo CA, 1988.

[34] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing : Explorations in The Microstructure of Cognition. Volume 1 : Foundations*. The MIT Press, 1986.

[35] B. Sallans. A hierarchical community of experts. Master's thesis, University of Toronto, Canada, 1998.

[36] L. K. Saul, T. Jaakkola, and M. I. Jordan. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4(4):61–76, 1996.

[37] L. K. Saul and M. I. Jordan. Attractor dynamics in feedforward neural networks. Submitted for publication.

[38] L. K. Saul and M. I. Jordan. A mean field learning algorithm for unsupervised neural networks. Technical report, Massachusetts Institute of Technology, 1997.

[39] E. Saund. Unsupervised learning of mixtures of multiple causes in binary data. In *Advances in Neural Information Processing Systems*, volume 6. Morgan Kaufmann Publishers, San Mateo CA, 1993.

[40] P. Simard, Y. Le Cun, and J. Denker. Efficient pattern recognition using a new transformation distance. In S. Hanson, J. Cowan, and L. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5. Morgan Kaufmann Publishers, San Mateo CA, 1992.

[41] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analysis. Technical Report NCRG/97/003, Aston University, Department of Computer Science and Applied Mathematics, 1997.

[42] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. Technical Report NCRG/97/010, Aston University, Department of Computer Science and Applied Mathematics, 1997.

[43] R. S. Zemel, M. C. Mozer, and G. E. Hinton. TRAFFIC: Recognizing objects using hierarchical reference frame transformations. In *Advances in Neural Information Processing Systems*, volume 2. Morgan Kaufmann Publishers, San Mateo CA, 1989.