Assignments 4 and 5

Probabilistic and Unsupervised Learning

Maneesh Sahani

Due: Thu Dec 4, 2014

Note: all assignments for this course are to be handed in to the Gatsby Unit, **not** to the CS department. Assignments are due at the beginning of the lecture or tutorial on the due date. Late assignments (included those handed in later on the due day) will be penalised. If you are unable to attend, you may hand in your assignment to a TA prior to the due time, or to Barry Fong in the Alexandra House 4th floor reception. Do not leave them with anyone else.

This is a combined assignment worth 150 marks (plus bonuses). You would be well advised to begin work early.

Please attempt the main questions before the bonus ones.

1. **[80 marks] Mean-field learning**

   Consider a binary latent factor model. This is a model with a vector $\mathbf{s}$ of K binary latent variables, $\mathbf{s} = (s_1, \ldots, s_K)$, a real-valued observed vector $\mathbf{x}$ and parameters $\boldsymbol{\theta} = \{\{\boldsymbol{\mu}_i, \pi_i\}_{i=1}^K, \sigma^2\}$. The model is described by:

$$p(\mathbf{s}|\boldsymbol{\pi}) = p(s_1, \ldots, s_K|\boldsymbol{\pi}) = \prod_{i=1}^{K} p(s_i|\pi_i) = \prod_{i=1}^{K} \pi_i^{s_i}(1-\pi_i)^{(1-s_i)}$$

$$p(\mathbf{x}|s_1, \ldots, s_K, \boldsymbol{\mu}, \sigma^2) = \mathcal{N}\left(\sum_i s_i \boldsymbol{\mu}_i, \sigma^2 I\right)$$

   where $\mathbf{x}$ is a D-dimensional vector and I is the $D \times D$ identity matrix. Assume you have a data set of N i.i.d. observations of $\mathbf{x}$, i.e. $\mathcal{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$.

   Matlab hint: Wherever possible, avoid looping over the data points. Many (but not all) of these functions can be written using matrix operations. In Matlab it's much faster.

   Warning: Each question depends on earlier questions. Start as soon as possible.

   Hand in: Derivations, code and plots.

   We will implement generalized EM learning using the fully factored (a.k.a. mean-field) **variational approximation** for the model above. That is, for each data point $\mathbf{x}^{(n)}$, we will approximate the posterior distribution over the hidden variables by a distribution:

$$q_n(\mathbf{s}^{(n)}) = \prod_{i=1}^{K} \lambda_{in}^{s_i^{(n)}} (1-\lambda_{in})^{(1-s_i^{(n)})}$$

   and find the $\boldsymbol{\lambda}^{(n)}$'s that maximize $\mathcal{F}_n$ holding $\boldsymbol{\theta}$ fixed.

   (a) Write a Matlab function:
   `[lambda,F] = MeanField(X,mu,sigma,pie,lambda0,maxsteps)`
   where `lambda` is $N \times K$, `F` is the lower bound on the likelihood, `X` is the $N \times D$ data matrix ($\mathcal{X}$), `mu` is the $D \times K$ matrix of means, `pie` is the $1 \times K$ vector of priors on $\mathbf{s}$, `lambda0` are initial values for `lambda` and `maxsteps` are maximum number of steps of the fixed point equations. You might also want to set a convergence criterion so that if `F` changes by less than some very small number ϵ the iterations halt. [20 marks]

(b) We have derived the M step for this model in terms of the quantities: `X`, `ES` $= E_q[\mathbf{s}]$, which is an $N \times K$ matrix of expected values, and `ESS`, which is an $N \times K \times K$ array of expected values $E_q[\mathbf{s}\mathbf{s}^\top]$ for each n. The full derivation is provided in Appendix B. Write two or three sentences discussing how the solution relates to linear regression and why. [5 marks]

(c) Using the above, we have implemented a function:

```
[mu, sigma, pie] = MStep(X,ES,ESS)
```

This can be implemented either taking in `ESS` = a $K \times K$ matrix summing over N the `ESS` array as defined above, or taking in the full $N \times K \times K$ array. This code can be found in Appendix C and can also be found on the web site. Study this code and figure out what the computational complexity of the code is in terms of N, K and D for the case where `ESS` is $K \times K$. Write out and justify the computational complexity; don't assume that any of N, K, or D is large compared to the others. [5 marks]

(d) Examine the data `images.jpg` shown on the web site (Do **not** look at `genimages.m` yet!). This shows 100 greyscale 4×4 images generated by randomly combining several features and adding a little noise. Try to guess what these features are by staring at the images. How many are there? Would you expect factor analysis to do a good job modelling this data? How about ICA? mixture of Gaussians? Explain your reasoning. [10 marks]

(e) Put the E step and M step code together into a function:

```
[mu, sigma, pie] = LearnBinFactors(X,K,iterations)
```

where K is the number of binary factors, and `iterations` is the maximum number of iterations of EM. Include a check that `F` increases at every iteration (this is a good debugging tool). [10 marks]

(f) Run your algorithm for learning the binary latent factor model on the data set generated by `genimages.m`. What features `mu` does the algorithm learn (rearrange them into 4×4 images)? How could you improve the algorithm and the features it finds? Explain any choices you make along the way and the rationale behind them (e.g. what to set K, how to initialize parameters, hidden states, and `lambdas`). [10 marks]

(g) For the setting of the parameters learned in the previous step, run the variational approximation for just the first data point (i.e. to find $q_1(\mathbf{s}^{(1)})$) (i.e. set $N = 1$). Convergence of a variational approximation results when the value of λ's as well as F stops changing. Plot `F` and `log(F(t)-F(t-1))` as a function of iteration number `t` for `MeanField`. How rapidly does it converge? Plot `F` for three widely varying `sigma`s. How is this affected by increases and decreases of `sigma`? Why? Support your arguments. [10 marks]

(h) Describe a (variational) Bayesian method for selecting K, the number of hidden binary variables in this model. Does your method pose any computational difficulties and if so how would you tackle them? [10 marks]

2. **[30 marks] EP for the binary factor model**

Now derive an EP algorithm to infer the marginals on the source variables in the same binary latent factor model.

(a) First, write down the log-joint probability for a single observation-source pair $\log(p(\mathbf{s}, \mathbf{x}))$. Rearrange the terms to form a sum of log-factors on $\mathbf{s}$ (assuming $\mathbf{x}$ is observed), each defined either on a single source variable, or on a pair:

$$\log(p(\mathbf{s}, \mathbf{x})) = \sum_i \log f_i(s_i) + \sum_{ij} \log g_{ij}(s_i, s_j).$$

Do the factors correspond to a standard exponential family form? Remember that, since the sources s are binary, $s_i^2 = s_i$.

(b) Next, derive a message passing scheme to find iterative approximations to each factor. Start your derivation from the KL divergence $\mathbf{KL}[p\|q]$ and identify clearly each time you make an approximate step. You don't need to make all of the EP approximations: which one(s) is(are) missing? Is this really an EP algorithm?

Give the final message-passing scheme in terms of updates to the natural parameters of the site approximations. There will be two different types of update: for the $\tilde{f}_i$ and the $\tilde{g}_{ij}$ respectively.

(c) Describe a Bayesian method for selecting K, the number of hidden binary variables using EP. Does your method pose any computational difficulties and if so how would you tackle them?

3. **[Bonus: 50 marks]** Implement the EP algorithm you derived in the previous question, and compare your results to those of the variational mean-field algorithm.

4. **[20 marks] EP for positivity constraints**

Consider a linear dynamical system:

$$y_1 \sim \mathcal{N}(0, \sigma^2) \tag{1}$$

$$y_i | y_{i-1} \sim \mathcal{N}(y_{i-1}, \sigma^2) \qquad \text{for } i = 2, 3, \ldots \tag{2}$$

$$x_i | y_i \sim \mathcal{N}(y_i, \tau^2) \qquad \text{for } i = 1, 2, \ldots \tag{3}$$

with each random variable being scalar. Suppose that we do not observe the exact values of the x_i's, but do observe that they are positive. We will now derive two different expectation propagation algorithms to approximate the resulting posterior over the y_i's.

(a) To incorporate the positivity observations, we could include additional factors of the form:

$$f_i(x_i) = \begin{cases} 1 & \text{if } x_i > 0, \\ 0 & \text{otherwise} \end{cases}$$

Derive an expectation propagation algorithm to estimate the marginal distributions over all x_i and y_i in the joint distribution given by the (normalized) product of these factors with the distribution of equations (1-3). Approximate each factor with a Gaussian. You may assume access to a function which can compute the mean $E(m, s^2)$ and variance $V(m, s^2)$ of the truncated Gaussian:

$$P(z|m, v) \propto \begin{cases} e^{-\frac{(z-m)^2}{2s^2}} & \text{if } z > 0; \\ 0 & \text{otherwise} \end{cases}$$

(b) An alternative approach would be to first compute the probabilities:

$$g_i(y_i) = P(x_i > 0 | y_i),$$

and then use expectation propagation to estimate the marginals of y_i's in the joint distribution given by the product of the g_i factors with the prior $P(y_1, \ldots, y_t)$ given in equations (1-2). Show that both EP algorithms are equivalent in that they should have the same fixed points.

5. **[15 marks] Posterior and Loopy Belief Propagation**

(a) For the model in Question 1, show that the posterior distribution over $\mathbf{s}$ given $\mathbf{x}$ can be expressed as a Boltzmann machine. [5 marks]

(b) Suppose instead of mean field inference, you wish to use loopy belief propagation for inference. Describe at a high level the changes to the approach in Question 1 to accommodate using loopy belief propagation. What would you use for $E_q[\mathbf{s}]$ and $E_q[\mathbf{s}\mathbf{s}^\top]$? Would the resulting approximate EM algorithm be guaranteed to converge?

6. **[5 marks] Binary MRF and Boltzmann Machines** In the max-cut/min-flow lecture we described a binary attractive MRF as a joint distribution over binary variables $X_i \in \{0, 1\}$ parametrized by:

$$p(\mathbf{X}) \propto \exp\left(\sum_{(ij)} W_{ij}\delta(X_i = X_j) + \sum_i c_i X_i\right)$$

Show that this distribution can also be parametrized by a Boltzmann machine. Describe how the parameters of the Boltzmann machine relate to the parameters of the binary attractive MRF above. Can any Boltzmann machine be parametrized as a (not necessarily attractive) binary MRF?

7. **[Bonus 5 marks] Inconsistency of Local Marginals** Loopy belief propagation approximates the distribution over a pairwise MRF using a set of locally consistent beliefs $\{b_i(x_i), b_{ij}(x_i, x_j)\}$:

$$\sum_{x_i} b_i(x_i) = 1 \qquad \text{for all } i;$$
$$\sum_{x_i} b_{ij}(x_i, x_j) = b_j(x_j) \qquad \text{for all } i, j \text{ and } x_j.$$

Give an example set of beliefs that are locally consistent but not globally consistent. That is, there is no distribution $p(\mathbf{X})$ over all variables such that

$$p(X_i = x_i) = b_i(x_i) \qquad \text{for all } i, x_i;$$
$$p(X_i = x_i, X_j = x_j) = b_{ij}(x_i, x_j) \qquad \text{for all } i, j, x_i, x_j.$$

Explain why this set of beliefs is not globally consistent. Hint: the MRF has to contain at least a loop since for tree-structured distributions local consistency implies global consistency.

8. **[Bonus 5 marks] Inconsistency of Local Marginals (contd)** Give an example of a graphical model with specific parameter settings, such that the local marginals you came up with in the previous question is a fixed point of the loopy belief propagation algorithm run on this model.

Appendix: M-step for Assignment [5]

Iain Murray

December 2003[1]

A Background

The generative model under consideration has a vector of K binary latent variables $\mathbf{s}$. Each D-dimensional data point $\mathbf{x}^{(n)}$ is generated using a new hidden vector, $\mathbf{s}^{(n)}$. Each $\mathbf{s}^{(n)}$ is identically and independently distributed according to:

$$P\left(\mathbf{s}^{(n)}|\boldsymbol{\pi}\right) = \prod_{i=1}^{K} \pi_i^{s_i^{(n)}} (1-\pi_i)^{(1-s_i^{(n)})}. \tag{4}$$

Once $\mathbf{s}^{(n)}$ has been generated, the data point is created according to the Gaussian distribution:

$$p\left(\mathbf{x}^{(n)}\middle|\mathbf{s}^{(n)}, \boldsymbol{\mu}, \sigma^2\right) = (2\pi\sigma^2)^{-D/2} \exp\left[-\frac{1}{2\sigma^2}\left(\mathbf{x}^{(n)} - \sum_{i=1}^{K} s_i^{(n)}\boldsymbol{\mu}_i\right)^{\top}\left(\mathbf{x}^{(n)} - \sum_{i=1}^{K} s_i^{(n)}\boldsymbol{\mu}_i\right)\right]. \tag{5}$$

When this process is repeated we end up obtaining a set of visible data $\mathcal{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ generated by a set of N binary vectors $\mathcal{S} = \{\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(N)}\}$ and some model parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \sigma^2, \boldsymbol{\pi}\}$, which are constant across all the data. Given just $\mathcal{X}$, both $\mathcal{S}$ and $\boldsymbol{\theta}$ are unknown. We might want to find the set of parameters that maximise the likelihood function $P(\mathcal{X}|\boldsymbol{\theta})$; "the parameters that make the data probable". EM is an approach towards this goal which takes our knowledge about the uncertain $\mathcal{S}$ into account.

In the EM algorithm we optimise the objective function

$$\begin{aligned}\mathcal{F}(q, \boldsymbol{\theta}) &= \langle \log p(\mathcal{S}, \mathcal{X}|\,\boldsymbol{\theta})\rangle_{q(\mathcal{S})} - \langle \log q(\mathcal{S})\rangle_{q(\mathcal{S})} \\ &= \sum_n \left\langle \log p\left(\mathbf{s}^{(n)}, \mathbf{x}^{(n)}\middle|\,\boldsymbol{\theta}\right)\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} - \sum_n \left\langle \log q\left(\mathbf{s}^{(n)}\right)\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}\end{aligned}, \tag{6}$$

alternately increasing $\mathcal{F}$ by changing the distribution $q(\mathcal{S})$ in the "E-step", and the parameters in the "M-step". This document gives a derivation and Matlab implementation of the M-step. In this assignment you will implement a variational E-step and apply this EM algorithm to a data set.

[1]Modified to match updated notation in 2006

B M-step derivation

Here we maximise $\mathcal{F}$ with respect to each of the parameters using differentiation. This only requires the term with $\boldsymbol{\theta}$ dependence:

$$\sum_n \left\langle \log p\left(\mathbf{s}^{(n)}, \mathbf{x}^{(n)} \middle| \boldsymbol{\theta}\right)\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} = \sum_n \left\langle \log p\left(\mathbf{x}^{(n)} | \mathbf{s}^{(n)}, \boldsymbol{\theta}\right) + \log P\left(\mathbf{s}^{(n)} | \boldsymbol{\theta}\right)\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} \tag{7}$$

Substituting the given distributions from equations 5 and 4 gives:

$$\begin{aligned}
&= -\frac{ND}{2}\log 2\pi - ND\log\sigma \\
&\quad -\frac{1}{2\sigma^2}\left[\sum_{n=1}^N \mathbf{x}^{(n)\top}\mathbf{x}^{(n)} + \sum_{i,j}\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_j \sum_{n=1}^N \left\langle s_i^{(n)} s_j^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} - 2\sum_i \boldsymbol{\mu}_i^\top \sum_{n=1}^N \left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} \mathbf{x}^{(n)}\right] \\
&\quad + \sum_{i=1}^K \left[\log \boldsymbol{\pi}_i \sum_{n=1}^N \left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} + \log\left(1-\boldsymbol{\pi}_i\right)\left(N - \sum_{n=1}^N \left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}\right)\right] .
\end{aligned} \tag{8}$$

From which we can obtain all the required parameter settings:

$$\frac{\partial\mathcal{F}}{\partial\pi_i} = \frac{1}{\pi_i}\sum_{n=1}^N \left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} + \frac{1}{1-\pi_i}\left[\sum_{n=1}^N \left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} - N\right] = 0 \tag{9}$$

$$\Rightarrow \boxed{\boldsymbol{\pi} = \tfrac{1}{N}\textstyle\sum_{n=1}^N \left\langle \mathbf{s}^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}} \; , \tag{10}$$

$$\begin{aligned}
\frac{\partial\mathcal{F}}{\partial\boldsymbol{\mu}_i} &= -\frac{1}{\sigma^2}\sum_{n=1}^N \left[\sum_j \left\langle s_i^{(n)} s_j^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} - \left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} \mathbf{x}^{(n)}\right] \\
\sum_j \sum_{n=1}^N \left\langle s_i^{(n)} s_j^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} \boldsymbol{\mu}_j &= \sum_{n=1}^N \left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} \mathbf{x}^{(n)}
\end{aligned} \tag{11}$$

$$\Rightarrow \boxed{\boldsymbol{\mu}_j = \textstyle\sum_i \left[\sum_{n=1}^N \left\langle \mathbf{s}^{(n)}\mathbf{s}^{(n)\top}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}\right]^{-1}_{ji} \sum_{n=1}^N \left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} \mathbf{x}^{(n)}} \tag{12}$$

and

$$\frac{\partial\mathcal{F}}{\partial\sigma} = 0 \Rightarrow \boxed{\begin{array}{c}\sigma^2 = \frac{1}{ND}\left[\sum_{n=1}^N \mathbf{x}^{(n)\top}\mathbf{x}^{(n)} + \sum_{i,j}\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_j \sum_{n=1}^N \left\langle s_i^{(n)} s_j^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}\right. \\ \left. -2\sum_i \boldsymbol{\mu}_i^\top \sum_{n=1}^N \left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} \mathbf{x}^{(n)}\right]\end{array}} \; . \tag{13}$$

Note that the required sufficient statistics of $q\left(\mathcal{S}\right)$ are $\left\langle \mathbf{s}^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}$ and $\sum_{n=1}^N \left\langle \mathbf{s}^{(n)}\mathbf{s}^{(n)\top}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}$. In the code these are known as `ES` and `ESS`.

All of the sums above can be interpreted as matrix multiplication or trace operations. This means that each of the boxed parameters above can neatly be computed in one line of Matlab.

C M-step code

MStep.m

```
% [mu, sigma, pie] = MStep(X,ES,ESS)
%
% Inputs:
% -----------------
%          X NxD data matrix
%         ES NxK E_q[s]
%        ESS KxK sum over data points of E_q[ss'] (NxKxK)
%               if E_q[ss'] is provided, the sum over N is done for you.
%
% Outputs:
% --------
%         mu DxK matrix of means in p(y|{s_i},mu,sigma)
%      sigma 1x1 standard deviation in same
%        pie 1xK vector of parameters specifying generative distribution for s
%

function [mu, sigma, pie] = MStep(X,ES,ESS)

[N,D] = size(X);
if (size(ES,1)~=N), error('ES must have the same number of rows as X'); end;
K = size(ES,2);
if (isequal(size(ESS),[N,K,K])), ESS = shiftdim(sum(ESS,1),1); end;
if (~isequal(size(ESS),[K,K]))
    error('ESS must be square and have the same number of columns as ES');
end;

mu = (inv(ESS)*ES'*X)';
sigma = sqrt((trace(X'*X)+trace(mu'*mu*ESS)-2*trace(ES'*X*mu))/(N*D));
pie = mean(ES,1);
```