4F13 Machine Learning: Coursework #2: Gibbs Sampling

Zoubin Ghahramani & Carl Edward Rasmussen

Due: 4pm Thursday Feb 18th, 2010 to Rachel Fogg, room BNO-37

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

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

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

where $\mathbf{y}$ 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 $\mathbf{y}$, i.e. $Y = \{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(N)}\}$. More details are provided in Appendix A.

General 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. Your solution should not exceed 6 pages.

a) 10% : Gibbs sampling relies on computing conditional probabilities of certain variables given all other variables (read section 29.5 in MacKay's textbook if you haven't done so already). Derive the conditional probability:

$$p(s_i^{(n)}|s_1^{(n)}, \ldots, s_{i-1}^{(n)}, s_{i+1}^{(n)}, \ldots, s_K^{(n)}, \mathbf{y}^{(n)}, \theta)$$

for the binary latent variable model, which you will need for Gibbs sampling. Describe an algorithm for drawing samples from this distribution, using the `rand` function in Matlab which gives you uniformly distributed random numbers between 0 and 1.

b) 30% : Using the conditional probability derived in the first question you will now implement **Gibbs sampling** to approximate the posterior distribution over the hidden states given the data. Specifically, write a function:

```
[S] = Gibbs(Y, mu, sigma, pie, S0, nsamples)
```

where `S` is a $N \times K \times$ `nsamples` array of samples over the hidden variables, `S0` is an $N \times K$ matrix of initial settings for the hidden variables.

c) 10% : We have derived the M step for this model in terms of the quantities: `Y`, `ES` $= E_q[s]$, 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 maximum likelihood linear regression and why.

d) 10% : Using the above, we have implemented a function:

```
[mu, sigma, pie] = MStep(Y,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.

e) 10% : 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 mixture of Gaussians? Explain your reasoning.

f) 30% : Make sure you understand the idea of *convergence* of Markov chains (29.8 and 29.9 in MacKay's book). Load the parameters from the `parameters.mat` file. Given just the first data point in the data set $\mathbf{y}^{(1)}$, i.e. $\mathsf{N} = 1$, run your Gibbs sampling code with various initial conditions of the hidden state `S0`. In particular, you might want to try starting with `S0` as all 1's, as all 0's, or somewhere in between. Plot various statistics of `S` (such as the sum over all `K` states, which should range between 0 and `K`, or the cumulative average over all `K` states) as a function of sampling iteration for Gibbs sampling for a large number of samples. Can you assess (visually) how long it takes for the Gibbs sampler to converge? How is this affected by increases and decreases in `sigma`? Why? Support your arguments.

g) 0% (optional) : Combine the Gibbs sampler you wrote in part b) with the `MStep` function to create an EM algorithm for learning in this model. Run run your algorithm on the full set of training data given the correct value of `K`. How well is the EM algorithm able to recover the parameters used to generate the data?

Appendix: M-step for Course work #2

Iain Murray, Dec 2003

A Background

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

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

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

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

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

In the EM algorithm we optimise the objective function

$$\begin{aligned}\mathcal{F}(q, \boldsymbol{\theta}) &= \langle \log p\,(S, Y|\,\boldsymbol{\theta})\rangle_{q(S)} - \langle \log q\,(S)\rangle_{q(S)} \\ &= \sum_n \left\langle \log p\left(\mathbf{s}^{(n)}, \mathbf{y}^{(n)}\middle|\,\boldsymbol{\theta}\right)\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} - \sum_n \left\langle \log q\left(\mathbf{s}^{(n)}\right)\right\rangle_{q\left(\mathbf{s}^{(n)}\right)},\end{aligned} \tag{3}$$

alternately increasing $\mathcal{F}$ by changing the distribution $q(S)$ 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 two approximate E-steps and apply this EM algorithm to a data set.

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:

$$\sum_n \left\langle \log p\left(\mathbf{s}^{(n)}, \mathbf{y}^{(n)}\middle|\,\boldsymbol{\theta}\right)\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} = \sum_n \left\langle \log p\left(\mathbf{y}^{(n)}|\mathbf{s}^{(n)}, \boldsymbol{\theta}\right) + \log P\left(\mathbf{s}^{(n)}|\boldsymbol{\theta}\right)\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} \tag{4}$$

Substituting the given distributions from equations 2 and 1 gives:

$$\begin{aligned}&= -\frac{ND}{2}\log 2\pi - ND\log\sigma \\ &\quad - \frac{1}{2\sigma^2}\left[\sum_{n=1}^{N} \mathbf{y}^{(n)\top}\mathbf{y}^{(n)} + \sum_{i,j} \boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j \sum_{n=1}^{N} \left\langle s_i^{(n)} s_j^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} - 2\sum_i \boldsymbol{\mu}_i^\top \sum_{n=1}^{N} \left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} \mathbf{y}^{(n)}\right] \\ &\quad + \sum_{i=1}^{K}\left[\log \boldsymbol{\pi}_i \sum_{n=1}^{N}\left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} + \log(1-\boldsymbol{\pi}_i)\left(N - \sum_{n=1}^{N}\left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}\right)\right].\end{aligned} \tag{5}$$

From which we can obtain all the required parameter settings:

$$\frac{\partial \mathcal{F}}{\partial \pi_i} = \frac{1}{\pi_i}\sum_{n=1}^{N}\left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} + \frac{1}{1-\pi_i}\left[\sum_{n=1}^{N}\left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} - N\right] = 0 \tag{6}$$

$$\Rightarrow \boxed{\boldsymbol{\pi} = \frac{1}{N}\sum_{n=1}^{N}\left\langle \mathbf{s}^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}} \;, \tag{7}$$

$$\frac{\partial \mathcal{F}}{\partial \boldsymbol{\mu}_i} = -\frac{1}{\sigma^2}\sum_{n=1}^{N}\left[\sum_j \left\langle s_i^{(n)} s_j^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} - \left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} \mathbf{y}^{(n)}\right]$$

$$\sum_j \sum_{n=1}^{N}\left\langle s_i^{(n)} s_j^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} \boldsymbol{\mu}_j = \sum_{n=1}^{N}\left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} \mathbf{y}^{(n)} \tag{8}$$

$$\Rightarrow \boxed{\boldsymbol{\mu}_j = \sum_i \left[\sum_{n=1}^{N}\left\langle \mathbf{s}^{(n)}\mathbf{s}^{(n)\top}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}\right]^{-1}_{ji} \sum_{n=1}^{N}\left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} \mathbf{y}^{(n)}} \tag{9}$$

and

$$\frac{\partial \mathcal{F}}{\partial \sigma} = 0 \Rightarrow \boxed{\begin{array}{c}\sigma^2 = \frac{1}{ND}\left[\sum_{n=1}^{N}\mathbf{y}^{(n)\top}\mathbf{y}^{(n)} + \sum_{i,j}\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_j\sum_{n=1}^{N}\left\langle s_i^{(n)} s_j^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}\right. \\ \left. -2\sum_i \boldsymbol{\mu}_i^\top \sum_{n=1}^{N}\left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} \mathbf{y}^{(n)}\right]\end{array}} \;. \tag{10}$$

Note that the required sufficient statistics of $q(S)$ 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(Y,ES,ESS)
%
% Inputs:
% -----------------
%        Y 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(Y,ES,ESS)

[N,D] = size(Y);
if (size(ES,1)~=N), error('ES must have the same number of rows as Y'); 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'*Y)';
sigma = sqrt((trace(Y'*Y)+trace(mu'*mu*ESS)-2*trace(ES'*Y*mu))/(N*D));
pie = mean(ES,1);
```