4F13 Machine Learning: Coursework #3: Variational Inference

Zoubin Ghahramani & Carl Edward Rasmussen

Due: 4pm Tuesday March 8th, 2011 to Rachel Fogg, room BNO-37

Consider the following binary latent factor model exactly as in Coursework 2, 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 $\boldsymbol{\theta} = \{\{\boldsymbol{\mu}_i, \pi_i\}_{i=1}^K, \sigma^2\}$. The model is described by:

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

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

where $\mathbf{y}$ us 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 7 pages.

a) 40% : In this exercise you will implement the fully factored (a.k.a. mean-field) **variational approximation** described in the course notes. That is, for each data point $\mathbf{y}^{(n)}$, the code 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. Specifically, you should write a Matlab function:

`[lambda,F] = MeanField(Y,mu,sigma,pie,lambda0,maxsteps)`

where `lambda` is $N \times K$, `F` is the lower bound on the likelihood, `Y` is the $N \times D$ data matrix, `mu` is the $D \times K$ matrix of means, `pie` is the $1 \times K$ vector of priors on `s`, `lambda0` are initial values for `lambda` and `maxsteps` are maximum number of steps of the fixed point equations. Run the function on data generated from `genimages.m` using the parameters in `parameters.mat` from Coursework 2, and plot the value of the lower bound as a function of steps. You might want to set a convergence criterion so that if `F` changes by less than some very small number ϵ the iterations halt.

b) 20% : Using the parameters in `parameters.mat` and given just the first data point in the data set $\mathbf{y}^{(1)}$, run the `MeanField` function in a). 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.

c) 20% : Recall from Coursework 2 the function:

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

Put the variational E step and this M step code together into a function:

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

where K is the number of binary factors, `iterations` is the maximum number of iterations of EM. Make sure `F` always increases (this is a good debugging tool). This function should start by initialising the parameters randomly from some suitable distribution. Describe what suitable means in this context.

d) 20% : Run your algorithm for learning the binary latent factor model on the data set generated by `genimages.m` for several random parameter initialisations. What features `mu` does the algorithm learn (rearrange them into 4×4 images) and how do they compare to the true parameters?