COMP0085: Summative Assignments

Probabilistic and Unsupervised Learning / Approximate Inference

Maneesh Sahani

1. **[40 marks] A biochemical pathway**

   A friend who works for a pharmaceutical company has asked for your help analysing the concentrations of 9 species of molecule within a biochemical cascade. The process is a trade secret, so she refers to them by the letters A – I. She tells you that:

   - The concentrations of both species B and C depend on that of A.
   - Molecule C seems to redirect the process that produces enzyme D to produce enzyme E instead.
   - D catalyses the production of F from B, while E catalyses the production of H from G.
   - F and H then combine to form the end product I.

   While these general facts are well established, the reaction kinetics are very sensitive to small temperature and stereochemical fluctations; making each link rather noisy. This means that the parameters have not been well characterised.

   (a) Using the information you friend has given you, draw a directed acyclic graph (DAG or Bayes Net) showing how the concentrations of the 9 species depend on one another.

   [5 marks]

   (b) Show:

   - the moralised graph
   - an efficient triangulation
   - the resulting junction tree
   - the junction tree redrawn as a factor graph.

   [15 marks]

   (c) Find the (non-unique) smallest set of molecules, such that if the concentrations of the species within the set are known, the concentrations of the others would all be independent (conditioned on the measured ones).

   [5 marks]

   Your friend has gathered data on the concentrations of some of the species over a number of experiments. She expresses each concentration as a perturbation around its mean value (the mean being calculated over a very large number of experiments). Let us write these concentration perturbations as δ[A] etc. She suggests that a reasonable parametric model for the dependence of each perturbation on that of its parents might be to take a weighted linear combination of the parent concentration perturbations and add a reaction-specific Gaussian noise variable.

   Unfortunately, the only measurements she is willing to give you are those of δ[B], δ[D], δ[E], and δ[G].

(d) Given the linear-Gaussian structure, you decide to try factor analysis on these measurements; that is, a model of the form

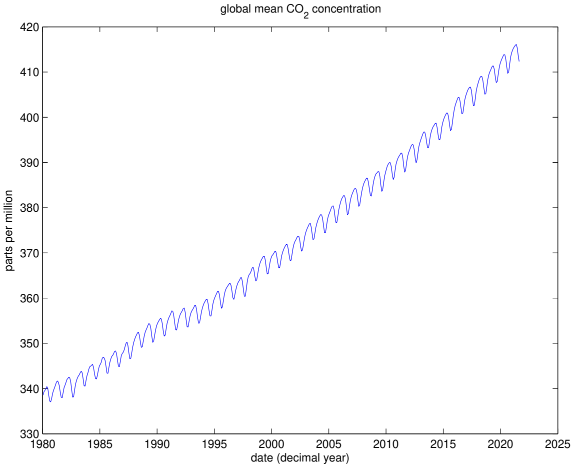
$$\mathbf{z} \sim \mathcal{N}(0, I)$$
$$\mathbf{x}|\mathbf{z} \sim \mathcal{N}(\Lambda\mathbf{z}, \Psi)$$

where the covariance Ψ is diagonal, but with possibly unequal diagonal variances.

Given the known graph, how many factors would you expect to recover? What would these correspond to? *[5 marks]*

(e) Can the results of the factor analysis be used to recover the concentration perturbations of any other species in the cascade? What about the linear weights in the graph? Discuss the identifiability of each node and each weight in your DAG — can its value can be determined upto an unknown scale factor (recall that factor analysis can never uniquely identify the scale of a latent parameter). *[10 marks]*

2. **[35 marks] Bayesian linear and Gaussian process regression.** The following time series of monthly mean global CO_2 concentrations can be obtained from the file `co2.txt` (data downloaded from http://www.esrl.noaa.gov/gmd/ccgg/trends):



We will apply Bayesian linear and Gaussian process regression to predict the CO_2 concentration $f(t)$ as a function of time t, given as the "decimal year" in the file.

(a) First we model the function using linear regression, that is, using the functional form

$$f(t) = at + b + \epsilon(t),$$

with i.i.d. noise residual $\epsilon(t) \sim \mathcal{N}(0, 1)$ and prior $a \sim \mathcal{N}(0, 10^2)$, $b \sim \mathcal{N}(360, 100^2)$. Compute and show the posterior mean and covariance over a and b given the CO_2 data. [10 marks]

(b) Let $a_{\text{MAP}}, b_{\text{MAP}}$ be the MAP estimate in the question above. The residual is the difference between the observed function values and the predicted mean function values

$$g_{\text{obs}}(t) = f_{\text{obs}}(t) - (a_{\text{MAP}}t + b_{\text{MAP}}),$$

where $f_{\text{obs}}(t)$ is the observed value of the CO_2 concentration at time t.
Plot $g_{\text{obs}}(t)$. Do you think these residuals conform to our prior over $\epsilon(t)$? State, with justifications, which characteristics of the residual you think do or do not conform to our prior belief. [5 marks]

(c) Write a function to generate samples drawn from a GP. Specifically, given a covariance kernel function $k(\cdot, \cdot)$ and a vector of input points $\mathbf{x}$, return a function $f(\mathbf{x})$ evaluated on the input points $\mathbf{x}$ drawn randomly from a GP with the given covariance kernel and with zero mean.

[10 marks]

(d) Test your function by plotting sample functions drawn from the following kernel, for various settings of the hyperparameters

$$k(s,t) = \theta^2 \left(\exp\left(-\frac{2\sin^2(\pi(s-t)/\tau)}{\sigma^2} \right) + \phi^2 \exp\left(-\frac{(s-t)^2}{2\eta^2} \right) \right) + \zeta^2 \delta_{s=t} \qquad (1)$$

Describe the characteristics of the drawn functions, and how the characteristics of the functions depend on the parameters. [5 marks]

(e) Suppose we were to consider modelling the residual function $g(t)$ using a zero mean GP with the covariance kernel above. Based on the plot of $g(t)$ and your explorations in the preceding part, what do you think will be suitable values for the hyperparameters of k? [5 marks]

(f) [**Bonus**] Extrapolate the CO_2 concentration levels to 2020 using the GP with covariance kernel k of eqn 1, and your chosen parameter values. Specifically, compute the predictive mean and variance of the residual $g(t)$ for every month between September 2007 and December 2020 given the observed residuals $g_{\text{obs}}(t)$. Plot the means and one standard deviation error bars of the extrapolated CO_2 concentration levels

$$f(t) = a_{\text{MAP}}t + b_{\text{MAP}} + g(t)$$

along with the observed CO_2 levels. Does the behaviour of the extrapolation conform to your expectations? How sensitive are your conclusions to settings of the kernel hyperparameters? [15 bonus marks]

(g) [**Bonus**] Why is the above procedure not fully Bayesian? How would we go about modelling $f(t)$ in a Bayesian framework? [5 bonus marks]

3. **[70 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. $\mathcal{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$.

Warning: Each question depends on earlier questions.

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 function of the form:

`[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]

4. **[Bonus 40 marks] Variational Bayes for binary factors**

   For the model of the previous question:

   (a) Derive a (variational) Bayesian hyperparameter optimisation algorithm to automatically determine K, the number of hidden binary variables in this model, following the approach discussed in lecture for factor analysis. [10 marks]

   (b) Implement your algorithm, and demonstrate results on the data from Question 1. Fit models with a range of maximum values for K (from half the true number to twice or more).

   Provide a figure showing the VB free energy of each model as a function of iteration; as well a measure of the effective number of factors at each point. Interpret the figure: does the ordering of free energies, and their relationship to the effective number of factors, make sense? [30 marks].

5. **[30 marks] EP for the binary factor model**

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

   (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:

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

   Relate your result to the Boltzmann Machine. [Remember that, since the sources s are binary, $s_i^2 = s_i$.] [5 marks]

   (b) Next, derive a message passing scheme to find iterative approximations $\tilde{f}_i$ and $\tilde{g}_{ij}$ 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?

   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. [10 marks]

   (c) Rewrite your message passing approximation to use factored approximate messages. Explain how this leads to a loopy BP algorithm. [5 marks]

   (d) 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? [10 marks]

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

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:

$$P\left(\mathbf{s}^{(n)}|\boldsymbol{\pi}\right) = \prod_{i=1}^{K} \pi_i^{s_i^{(n)}} (1-\pi_i)^{(1-s_i^{(n)})}. \tag{2}$$

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

$$p\left(\mathbf{x}^{(n)}\middle|\,\mathbf{s}^{(n)}, \boldsymbol{\mu}, \sigma^2\right) = (2\pi\sigma^2)^{-D/2} \exp\left[-\frac{1}{2\sigma^2}\left(\mathbf{x}^{(n)} - \sum_{i=1}^{K} s_i^{(n)}\boldsymbol{\mu}_i\right)^{\top}\left(\mathbf{x}^{(n)} - \sum_{i=1}^{K} s_i^{(n)}\boldsymbol{\mu}_i\right)\right]. \tag{3}$$

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

$$\begin{aligned}\mathcal{F}(q,\boldsymbol{\theta}) &= \langle \log p\left(\mathcal{S},\mathcal{X}|\,\boldsymbol{\theta}\right)\rangle_{q(\mathcal{S})} - \langle \log q\left(\mathcal{S}\right)\rangle_{q(\mathcal{S})} \\ &= \sum_n \left\langle \log p\left(\mathbf{s}^{(n)}, \mathbf{x}^{(n)}\middle|\,\boldsymbol{\theta}\right)\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} - \sum_n \left\langle \log q\left(\mathbf{s}^{(n)}\right)\right\rangle_{q\left(\mathbf{s}^{(n)}\right)},\end{aligned} \tag{4}$$

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:

$$\sum_n \left\langle \log p\left(\mathbf{s}^{(n)}, \mathbf{x}^{(n)} \middle| \boldsymbol{\theta}\right)\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} = \sum_n \left\langle \log p\left(\mathbf{x}^{(n)} | \mathbf{s}^{(n)}, \boldsymbol{\theta}\right) + \log P\left(\mathbf{s}^{(n)} | \boldsymbol{\theta}\right)\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} \tag{5}$$

Substituting the given distributions from equations 3 and 2 gives:

$$\begin{aligned}= &-\frac{ND}{2}\log 2\pi - ND\log\sigma \\ &-\frac{1}{2\sigma^2}\left[\sum_{n=1}^{N}\mathbf{x}^{(n)\top}\mathbf{x}^{(n)} + \sum_{i,j}\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_j \sum_{n=1}^{N}\left\langle s_i^{(n)} s_j^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} - 2\sum_i \boldsymbol{\mu}_i^\top \sum_{n=1}^{N}\left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}\mathbf{x}^{(n)}\right] \\ &+\sum_{i=1}^{K}\left[\log\boldsymbol{\pi}_i \sum_{n=1}^{N}\left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} + \log\left(1-\boldsymbol{\pi}_i\right)\left(N - \sum_{n=1}^{N}\left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}\right)\right] \quad .\end{aligned} \tag{6}$$

From which we can obtain all the required parameter settings:

$$\frac{\partial\mathcal{F}}{\partial\pi_i} = \frac{1}{\pi_i}\sum_{n=1}^{N}\left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} + \frac{1}{1-\pi_i}\left[\sum_{n=1}^{N}\left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} - N\right] = 0 \tag{7}$$

$$\Rightarrow \boxed{\boldsymbol{\pi} = \tfrac{1}{N}\textstyle\sum_{n=1}^{N}\left\langle \mathbf{s}^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}} \quad , \tag{8}$$

$$\begin{aligned}\frac{\partial\mathcal{F}}{\partial\boldsymbol{\mu}_i} &= -\frac{1}{\sigma^2}\sum_{n=1}^{N}\left[\sum_j\left\langle s_i^{(n)} s_j^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} - \left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}\mathbf{x}^{(n)}\right] \\ \sum_j\sum_{n=1}^{N}\left\langle s_i^{(n)} s_j^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}\boldsymbol{\mu}_j &= \sum_{n=1}^{N}\left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}\mathbf{x}^{(n)}\end{aligned} \tag{9}$$

$$\Rightarrow \boxed{\boldsymbol{\mu}_j = \textstyle\sum_i\left[\sum_{n=1}^{N}\left\langle \mathbf{s}^{(n)}\mathbf{s}^{(n)\top}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}\right]_{ji}^{-1}\sum_{n=1}^{N}\left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}\mathbf{x}^{(n)}} \tag{10}$$

and

$$\frac{\partial\mathcal{F}}{\partial\sigma} = 0 \Rightarrow \boxed{\begin{array}{c}\sigma^2 = \frac{1}{ND}\left[\sum_{n=1}^{N}\mathbf{x}^{(n)\top}\mathbf{x}^{(n)} + \sum_{i,j}\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_j\sum_{n=1}^{N}\left\langle s_i^{(n)} s_j^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}\right. \\ \left. -2\sum_i\boldsymbol{\mu}_i^\top\sum_{n=1}^{N}\left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}\mathbf{x}^{(n)}\right]\end{array}} \quad . \tag{11}$$

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);
```