Assignment 5

Probabilistic and Unsupervised Learning

Maneesh Sahani

Due: Noon, Monday Jan 13, 2020

Note: As usual, please attempt the main questions before the bonus ones.

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

   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 `lambda`s). [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]

[Continued ...]

2. **[30 marks] Noisy ICA** Consider a noisy independent factor model for data $\mathbf{x} \in \mathbb{R}^D$:

$$\begin{aligned} \mathbf{x} &\sim \mathcal{N}(A\mathbf{z}, \Psi) && A \in \mathbb{R}^{D\times K}; \quad \Psi \in \mathbb{R}^{D\times D}, \text{ diagonal.} \\ z_k &\sim \mathcal{N}\left(0, u_k^{-1}\right) && k = 1 \dots K \\ u_k &\sim \mathsf{Gamma}\left(\frac{\alpha}{2}, \frac{\alpha}{2}\right) && k = 1 \dots K \end{aligned}$$

(a) Derive the resulting marginal $P(z_k)$ (integrating out u_k). What distribution is this? Sketch the density function and explain the relationship between this model and ICA. [5 marks]

This two-stage representation of the prior on z_k is valuable because it allows us to design approximate methods for inference in noisy ICA. For example, consider variational Bayes (VB).

(b) Write down a *minimal* VB factorisation of the joint posterior on latents and parameters that would make approximate learning tractable. By minimal we mean that you should not assume any factorisations beyond those necessary to obtain a tractable algorithm. [5 marks]

(c) Derive the update for the term $q(\Psi, \dots)$ (where the dots stand for any other variables that may be included in the factor according to your factorisation). You may assume conjugate priors wherever needed. [5 marks]

(d) Derive hyperparameter optimisation rules to:

(i) learn the shape of the marginal distributions $P(z_k)$;

(ii) learn the number of factors.

You may need to alter the model parameterisation in the second case. Explain the reasoning behind your choice of hyperparametrisation, and derive the corresponding update rules. [15 marks]

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

Return to the model of Question 1.

(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].

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:

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

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

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

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

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

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:

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

Substituting the given distributions from equations 2 and 1 gives:

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

From which we can obtain all the required parameter settings:

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

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

$$\frac{\partial\mathcal{F}}{\partial\boldsymbol{\mu}_i} = -\frac{1}{\sigma^2}\sum_{n=1}^{N}\left[\sum_j\left\langle s_i^{(n)}s_j^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)} - \left\langle s_i^{(n)}\right\rangle_{q\left(\mathbf{s}^{(n)}\right)}\mathbf{x}^{(n)}\right] \tag{8}$$

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

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

and

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

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);
```