Probabilistic Modelling and Reasoning: Assignment 2
School of Informatics, University of Edinburgh

Instructor: Dr Chris Williams

Handed out 22 November 2005. Due in 9 December 2005 by 4pm

Remember that plagiarism is a university offence. Please read the policy at http://www.inf.ed.ac.uk/admin/ITO/DivisionalGuidelinesPlagiarism.html.

Marking Breakdown

A results/answer correct plus extra achievement at understanding or analysis of results. Clear explanations, evidence of creative or deeper thought will contribute to a higher grade.

B results/answer correct or nearly correct and well explained.

C results/answer in right direction but significant errors.

D some evidence that the student has gained some understanding, but not answered the questions properly.

E/F/G serious error or slack work.

0 Mechanics

You should submit this assignment manually to the ITO office in Appleton Tower by the deadline. Handwritten paper submissions are acceptable if the handwriting is neat and legible.

Late submissions: The policy of the School of Informatics (as of 3 November 2004) is that a 5% penalty per (working) day for 5 days will be applied, after which an award of 0% will be made (in the absence of medical or other explanatory evidence).

I have included some notes at the end of the assignment that tell you some useful things about MATLAB. These are to be read in conjunction with the handout “Introduction to MATLAB” available from the PMR webpage.

1 Digit Classification Using PPCA [40%]

Load the data file digdat.mat into MATLAB. It contains four objects, X2tr, X2te, X3tr, X3te. Each of these is a 500 × 196 matrix. Each contains 500 different examples of a handwritten digit. Each digit is a 14 × 14 array of pixel values unfolded into a vector of length 196. There are 500 training images of 2s (X2tr), 500 testing images of 2s (X2te), 500 training images of 3s (X3tr), and 500 testing images of 3s (X3te). (These data are extracted from the MNIST dataset, see http://yann.lecun.com/exdb/mnist/.) To view the image corresponding to row 11 of X2tr, do
This plots the row as a 14 × 14 grayscale image (and makes the axes square so it looks nice).

Your job is to model each of the two training datasets with a probabilistic principal components analysis (PPCA) model, and then use these learned models to classify the data in the test sets. Recall that for PPCA, the model of the data is

\[ x = \mu + Wz + e \]

where \( z \sim N(0, I_m) \), \( W \) is a general \( p \times m \) matrix called the factor loadings matrix, and the noise term \( e \) is independent of \( z \) and has the form \( e \sim N(0, \sigma^2 I) \). Thus it has mean \( \mu \) and covariance \( C = WW^T + \sigma^2 I \). Use a \( m = 3 \)-dimensional latent space for modelling both the 2s and the 3s.

The netlab toolbox contains the function `ppca` which should be helpful in fitting the PPCA model to data. You can make use of this toolbox by giving the command

```
addpath /home/ckiw/matlab/netlab3.2/
```

within MATLAB on the .inf system. You can also download the code from http://www.ncrg.aston.ac.uk/netlab/down.php. Doing help ppca within MATLAB will bring up the following explanation of the function.

\[ [\text{VAR}, \text{U}, \text{LAMBDA}] = \text{PPCA}(X, \text{PPCA\_DIM}) \] computes the principal component subspace \( U \) of dimension \( \text{PPCA\_DIM} \) using a centred covariance matrix \( X \). The variable \( \text{VAR} \) contains the off-subspace variance (which is assumed to be spherical), while the vector \( \text{LAMBDA} \) contains the variances of each of the principal components. This is computed using the eigenvalue and eigenvector decomposition of \( X \).

You can also download the netlab help files from the url above. You may also find the MATLAB functions `mean` and `cov` useful.

**Question 1:**

- Show a plot of the mean 2 and the mean 3 from the \( X_{2tr} \) and \( X_{3tr} \) respectively, displayed as 14 × 14 images. Clearly label the grayscale axis being used in each image.

- Show plots of the three columns of \( W \) for the \( X_{2tr} \) data, displayed as 14 × 14 images. Clearly label the grayscale axis being used in each image. Explain what kinds of variability are explained by adding a multiple of each of these vectors to the mean 2.

- Give MATLAB code fragments that show how you generated the plots above.

Now use these learned models as classifiers. The PPCA model for 2s is a Gaussian with some mean and covariance structure, and similarly for the 3s. Write a MATLAB function `logprob(mean,C,x)` that takes as input the `mean` and covariance `C` of a PPCA Gaussian and a vector `x`, and returns the log probability \( \log p(x | \mu, C) \). Compute \( \log p(x | \mu_2, C_2) \) and \( \log p(x | \mu_3, C_3) \) and thus classify the digits in \( X_{2te} \) and \( X_{3te} \). HINT: Given the eigenvalues and variance returned by the `ppca` function it is easy to compute the log determinant of a PPCA covariance matrix. Recall that the determinant of a covariance matrix is the product of its eigenvalues.
Question 2:

- Provide a listing of your function `logprob`, and a short explanation of how it works.

- Give the $2 \times 2$ confusion matrix for this classifier (i.e. the number of 2s classified as 2s, the number of 2s classified as 3s, the number of 3s classified as 2s, and the number of 3s classified as 3s). Explain how you computed this confusion matrix. Comment on the classifier and its performance.

Fun stuff:
For a given PPCA model you can explore each of the factors by running a loop that adds a multiple $\alpha$ of one factor (a column of $W$) to the mean vector; note that $\alpha$ can be negative as well as positive. Thus you can synthesize a sequence of images and observe the effect of that factor on the mean. This may help you answer part of Question 1.

2 Fitting Mixture Models [30%]

In this section we will make use of the data in `digdatpr.mat`. This contains four objects, `Xpr2tr`, `Xpr2te`, `Xpr3tr`, `Xpr2te`. Each of these is a $500 \times 60$ matrix. Each contains the same digits as we used in section 1, but projected down onto the first 60 principal components (PCs) of all of the data. The first 60 PCs account for over 95% of the variation in the data. The mean $\mu$ and projection matrix $evec$ can be found in `proj.mat`.

We will fit Gaussian mixture models (GMMs) to the 60-d data, using the functions `gmm` and `gmmem` in the Netlab toolbox. Consider the MATLAB fragment:

```matlab
dim = 60;
ncomp = 3; % number of components
mix = gmm(dim, ncomp, 'diag')
options=foptions;
options(14)=5; % 5 iterations of kmeans to initialise gmm
seed = 0;
rand('state',seed); % k-means initializations uses a call to rand
mix = gmminit(mix, Xpr2tr, options);
options(1) = 1; % Prints out error values.
niter = 100; % number of iterations of EM to use for fitting
options(14)=niter;
[mix, options, errlog] = gmmem(mix, Xpr2tr, options);
```

The `gmm` call creates a GMM in `dim` dimensions with `ncomp` components and a diagonal covariance type specified by 'diag'. (I recommend you use this covariance type for the experiments below.) `gmminit` initializes the mixture model using the $k$-means algorithm. As $k$-means uses a random initialization you can set the seed to obtain different initializations.  

`gmmem` trains the GMM using the EM algorithm and returns the trained model `mix`, the updated `options` vector, and `errlog` contains the negative log likelihood ($-\mathcal{L}$) of the training data at each iteration.

Do `help gmm`, `help gmminit` and `help gmmem` for more information. You can also try the Netlab demos `demgmm1`, `demgmm2`, `demgmm3`, `demgmm4`, `demgmm5` to see Gaussian mixture models in action.

---

Note that this is not really the proper way to use a random number generator (RNG). A RNG produces a pseudo-random sequence from repeated calls to it without resetting the seed. However, setting the seed to different values is a convenient way to obtain different initializations in a reproducible fashion.
Question 3:

- Experiment with fitting GMMs to the data Xpr2tr. Vary ncomp from 1 to 5 and for each value of ncomp experiment with different values of seed. Choose niter suitably so you think convergence has been reached. Provide a table of your results. Comment on the results you obtain.

- Write a short MATLAB function to compute the log likelihood of a dataset under a given mixture model. HINT: look at gmmem; most of what you need is there. Hand in a printout of this function.

- Now evaluate the log likelihood of the test data Xpr2te under the mixture models you learned above, using your function. Provide a table of your results for both training and test sets and a suitable plot. Comment on these results. How might you choose the best GMM for Xpr2te from the ones learned above?

- Comment on why it makes sense to use diagonal rather than either spherical or full covariances on this dataset.

Note that you can visualize the cluster means in the original data space by using evec and adding on mu.

3 ICA [30%]

We consider noiseless square ICA, so that we have

\[ x = As, \quad s = Wx, \]

where \( W = A^{-1}, \) \( A \) and \( W \) are square, non-singular matrices, \( x \) is the data vector and \( s \) is the vector of sources. We assume that the sources are independent, i.e. that \( p_s(s) = \prod_{i=1}^{D} p_s(s_i). \)

NOTE: This is the standard ICA notation. Unfortunately there is a conflict with the lecture notes and handout where \( W \) is used differently. In this question we will use the standard ICA notation as defined above.

The question below makes use of the FastICA MATLAB toolbox available from http://www.cis.hut.fi/projects/ica/fastica/. I have downloaded the toolbox and you can use it by giving the command

```
addpath /home/ckiw/matlab/FastICA_25/
```

within MATLAB on the .inf system. Some documentation is available from the website and from the help available within MATLAB. You can fit an ICA model to the data Xpr2tr using the call

```
[icasig,A,W] = fastica(Xpr2tr', 'approach', 'symm', 'g', 'tanh', 'verbose', 'off');
```

This function returns the matrices \( A \) and \( W \) and also \( icasig, \) which contains the estimated sources for each data vector. Don’t worry too much about what the options mean, but note that there are a choice of densities we can assume for \( p_s(s). \) The \texttt{tanh} option corresponds to

\[ p_s(s) = \frac{1}{\pi} \text{sech}(s) = \frac{1}{\pi \cosh(s)} = \frac{2}{\pi(e^s + e^{-s})}. \]
Question 4:

- Assuming the training data \{x_i\}, \(i = 1, \ldots, n\), is i.i.d., write down an expression for the log-likelihood \(\mathcal{L}\) as a function of \(W\). HINT: for such models, the distribution of the data vectors may be expressed as \(p_x(x) = |W|p_s(Wx)\), where \(|W|\) denotes the determinant of \(W\).

One way to train an ICA model is to obtain the gradient \(\partial \mathcal{L}/\partial w_{ij}\) for each entry in the matrix \(W\) and then use gradient ascent. Derive \(\partial \mathcal{L}/\partial w_{ij}\), showing your working. You may use without proof the fact that \(\partial \log |W|/\partial W = (W^{-1})^T\), where \(\partial f/\partial W\) denotes the matrix of derivatives with element \(ij\) being \(\partial f/\partial w_{ij}\).

- Plot a graph of the assumed source density \(p_s(s) = \frac{1}{\pi} \text{sech}(s)\). On the same graph plot a Gaussian with zero mean and the same variance as \(p_s(s)\). (The variance of \(p_s(s) = \pi^2/4\).) Comment on the differences between these two distributions.

- If this ICA model is correct then the estimated sources should be drawn from the \(p_s(s)\) distribution. Create a histogram of the icasig data and compare it to \(p_s(s)\).

- Using the formula \(p_x(x) = |W|p_s(Wx)\) calculate the log likelihood of the data Xpr2tr under the learned ICA model and compare it to your results for the GMM.

4 Notes on Matlab

- Remember, there are only a limited number of licences for MATLAB. After you have finished using MATLAB, quit from the MATLAB session so that others can work.

- You can find out more about NETLAB functions (and many MATLAB functions) by typing `help` followed by the function name. Also, you can find the `.m` file corresponding to a given function using the `which` command, for example

  ```matlab
  >> which gmm
  /home/ckiw/matlab/netlab3.2/gmm.m
  ```

  An easy way to get a listing of a function is to give the command `type <function>`, e.g. `type gmm`.

- `close all` closes all the windows. It helps if things get cluttered.

- The command `clf` is very useful; it clears the current figure. This is useful if you have `hold on` set, which means that successive plots are plotted on top of each other. If this has become confusing, `clf` clears the figure so you can start again. `clear` clears your workspace of variables (check that the command `who` returns nothing).

- The command `axis('equal')` sets the aspect ratio so that equal tick mark increments on the x-,y-axis are equal in size. This makes circles plot as circles, instead of ellipses.

- Making plots. Read about this in the “Introduction to MATLAB”. Recall that a figure can be save as an encapsulated postscript file `myplot.eps` using

```matlab
print -deps2 myplot.eps
```