# Probabilistic Modelling and Reasoning Assignment 2

**Instructor:** Prof. Chris Williams

**Published:** Fri November 4 2011, Revised Nov 18
**Due:** Fri December 2, 2011 by 4pm

Remember that plagiarism is a university offence. Please read the policy at `http://www.ed.ac.uk/schools-departments/academic-services/students/postgraduate-taught/discipline/plagiarism` .

## 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** Serious error or slack work.

## 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.

The policy stated in the School of Informatics MSc Degree Guide is that normally you will not be allowed to submit coursework late. For exceptions to this, e.g. in case of serious medical illness or serious personal problems, see `http://www.inf.ed.ac.uk/student-services/teaching-organisation/for-taught-students/coursework-and-projects/late-coursework-submission` .

## 1 Principal Component Analysis [10%]

Download the assignment's software tarball from the course webpage and unpack it using the command `tar -xvf a211.tar`. The MATLAB file `script.m` first executes `addtoolboxes.m` which loads the toolboxes that you'll need to complete this assignment. These toolboxes include Prof. Neil Lawrence's motion capture toolbox[1], Prof. Ian Nabney's Netlab toolbox[2] and a slightly modified version of Prof. Zoubin Ghahramani's linear dynamical system toolbox[3].

The script then loads a Biovision hierarchical motion capture file (.bvh) from the `data/` directory into 3 MATLAB variables: `skeleton`, `sequence_X`, and `frame_length`. `skeleton` stores information about the actor's body (e.g., lengths of limbs) and `sequence_X` is a matrix of size $581 \times 69$ that stores information about the actor's movements in the motion capture sequence. Each row contains one frame of the motion capture sequence, and the columns represent the skeleton's joint angles in the different frames. The sequence can be visualised using the `skelPlayData` function:

```
skelPlayData(skeleton, sequence_X, frame_length);
```

We wish to represent the data in a lower-dimensional space in order to extract information about the properties of the actor's gait. We will first apply Principal Component Analysis (PCA) as a dimensionality reduction technique.

1. Write a function that computes the principal components of the data:

   ```
   [ mu, E, lambda, p ] = getEigenvectors(sequence)
   ```

   The function receives as input the sequence data and returns the mean pose `mu` as a column vector, and a matrix `E` whose columns are the eigenvectors corresponding to the (descending) ordered eigenvalues `lambda` of the covariance matrix of `sequence`. The additional output `p` is the cumulative percentage of the variance explained by each of the eigenvalues in `lambda`. Hand in the MATLAB code of the function.

   HINT: You will probably find the Matlab functions `eig`, `sort` and `cumsum` helpful.

2. Using the function from the previous question compute the eigenvalues and eigenvectors of the data. State the first two eigenvalues. Produce a plot that shows the cumulative variance as a function of the number of components and determine how many components are needed to explain at least 95% of the variance in the data.

3. We wish to have an intuition of the kind of variability explained by the principal components of the data. For this, we can visualize $\boldsymbol{\mu} + z\mathbf{e}_i$ for various values of $z$, where $\boldsymbol{\mu}$ is the mean of the data and $\mathbf{e}_i$ is the $i$-th principal component.

   For each component $\mathbf{e}_i$, we can generate a new motion sequence `sequence_Y`, in which each frame $j$ of the sequence is set to equal $\boldsymbol{\mu} + z_j\mathbf{e}_i$. Create two sequences (each

[1] http://staffwww.dcs.shef.ac.uk/people/N.Lawrence/mocap/

[2] http://www1.aston.ac.uk/eas/research/groups/ncrg/resources/netlab/

[3] http://mlg.eng.cam.ac.uk/zoubin/software.html

of length 581 frames) for the first two principal components $\mathbf{e}_1$ and $\mathbf{e}_2$, by varying $z$ linearly between appropriate values of $z^{\min}$ and $z^{\max}$.

Hand in the code that carries out this computation. Comment on the type of information captured by each of the principal components. You may also wish to use some well-chosen figures.

4. The projection of a frame $\mathbf{x}_j$ onto the $i$-th eigenvector $\mathbf{e}_i$ is given by $\mathbf{e}_i^T(\mathbf{x}_j - \boldsymbol{\mu})$. Calculate the projection of each frame of the original sequence `sequence_X` onto the first 2 principal components. Hand in the code that carries out this computation. Plot the projections in 2D space using MATLAB's `line` function. Comment on the structure of the projections. What does this plot tell us about the original sequence?

## 2 Generative Topographic Mapping [20%]

The Generative Topographic Mapping (GTM) model of Bishop, Svensen and Williams (1997) is described in the tech report NCRG_96_015 available from the course website under week 7. However, you do not need to read and understand the whole paper to do the question below. The basics were covered in the handout *Factor Analysis and Beyond* and in the lectures. Note that the mapping from the latent space to the dataspace is implemented as a radial basis function (RBF) network, so that the mapping from $\boldsymbol{z}$ to $x_i$, the $i$th dimension of $\boldsymbol{x}$, is given as $\phi_i(\boldsymbol{z}) = \sum_j w_{ij}\psi_j(\boldsymbol{z})$, where the $\psi$ functions are Gaussian radial basis functions in the Netlab implementation. When fitting the model to data its fit is initalized to the PCA solution.

1. We wish to fit a GTM model to the data in `sequence_X`. The function `gtm1dinittrain.m` initializes a GTM model with a 1D latent space and performs EM training. It shows the projection of the data, the locations of the centres of the Gaussians in the GTM and spheres of equal variance onto the first two principal components of the data. This is achieved by passing it the matrix `E` computed above.

   Using this function, train a 1D GTM model of `sequence_X` with 50 latent sample points and 7 RBF centres for 200 EM iterations. Comment on the structure of the GTM fit. Why might this fit be inappropriate as a 1D latent variable model of `sequence_X`? How do you think GTM should ideally fit to this dataset, perhaps obtained using a different initialization?

2. Using the Netlab function `gtmprob` compute the log likelihood of the data under the trained model. Report the result and hand in the code that carries out this computation.

3. The fit achieved by the GTM model is sensitive to the initial locations and variance of the Gaussians, and also to `num_rbf_centres` and `num_latent_points`. Obtain some different fits by varying these two parameters and compare the data log likelihoods to those given by the fit of the GTM model above. You may also wish to include a couple of plots to illustrate the fits obtained. Discuss the relationship you observe between the quality of the fit and the log likelihood.

4. The function `gtm2dinittrain.m` initializes a GTM model with a 2D latent space and performs EM training. Using this function, train a 2D GTM model of data in `sequence_X` with 50 latent sample points per dimension and 10 RBF centres per dimension for 50 EM iterations. Using the Netlab function `gtmlmean` compute the mean responsibility for each frame in 2D latent-space. Plot the responsibilities in 2D space using MATLAB's `line` function. How does the GTM projection differ from the PCA projection?

## 3 Factor Analysis [35%]

Factor Analysis (FA) models the data $\boldsymbol{x} \in \mathbb{R}^D$ using a set of latent variables $\boldsymbol{z} \in \mathbb{R}^M$ where $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_M)$ and $\boldsymbol{x} \sim \mathcal{N}(W\boldsymbol{z} + \boldsymbol{\mu}, \Psi)$. $D$ is the dimensionality of the observed data (69 in this case), $M$ the number of latent states, and $\Psi$ is a diagonal matrix.

Using the function `fa` by Dr. David Barber which is in the LDS toolbox, fit a FA model with a 2D latent space to the data. Use a maximum number of 50 iterations.

1. Calculate the mean of the posterior projection of each frame of the original sequence `sequence_X` into the FA model's 2D latent space (see §12.2.4 in Bishop, 2006). Store the projections in a $581 \times 2$ matrix `sequence_Z_FA`. Hand in the code that carries out this computation. Plot the projections in 2D space using MATLAB's `line` function. Comment on the structure of the projections. How do these projections compare to those generated by PCA?

2. (This question has been deleted.)

3. Above we have explored PCA as a method to reduce the dimensionality of the data. Why would it not make sense to first apply PCA to the data and then FA? Consider the effect that PCA would have on the data and the kind of structure FA attempts do discover.

4. FA models the data marginally as a Gaussian $\mathcal{N}(\boldsymbol{\mu}, C)$ where the covariance $C = WW^T + \Psi$. Thus, one way to obtain the likelihood of a datapoint $\mathbf{x}$ under the learned FA model would be to compute the covariance matrix and to then just compute the Gaussian likelihood. This gives

$$\log p(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T C^{-1}(\boldsymbol{x} - \boldsymbol{\mu}) - \frac{1}{2}\log|C| - \frac{D}{2}\log(2\pi). \quad (1)$$

   Using the above equation write a Matlab function `L = fallikelihood(X, W, Psi, mu)` that computes the log likelihood of a dataset under a FA model. The function takes as input the $D \times N$ dimensional matrix with the dataset `X`, the $D \times M$ factor loadings matrix `W`, a $D \times 1$ dimensional vector `Psi` that contains the elements on the diagonal of the noise variance $\Psi$ and the $D \times 1$ dimensional mean vector `mu`. Hand in the code for your function.

   HINT: `fa.m` contains code to calculate the log likelihood of a dataset under a FA model.

5. When fitting a FA model we need to choose the number of latent factors $M$ as a parameter. In order to choose $M$ for the data at hand perform 7-fold cross validation. For this purpose shuffle the frames randomly (you can use the Matlab function `randperm` for this purpose). Split your dataset into 7 batches of size 83. Then fit FA models with $M = 1, 5, 10, 15, 20, 25$. For each value of $M$ repeat the fitting 7 times, using 6 of the 7 batches as training data and the seventh batch as test data (not used for training!). For each model evaluate the log likelihood of the training data and of the test data under the learned model. Plot your results in a suitable manner and comment. Also explain why it makes sense to shuffle the frames before splitting the data into batches. Give one alternative method to cross validation for performing model selection, and explain briefly how it works.

6. Computation of the log likelihood as defined in equation 1 is extremely expensive when $D$ is large, since the covariance matrix is of size $D \times D$. We can use matrix identities to reduce the cost. Let a nonsingular matrix $H$ be written as $H = A + BDE$. Then we have the identity $H^{-1} = A^{-1} - A^{-1}B(D^{-1} + EA^{-1}B)^{-1}EA^{-1}$. Applying this to $WI_MW^T + \Psi$ we obtain

$$\boldsymbol{d}^T C^{-1} \boldsymbol{d} = \boldsymbol{d}^T \Psi^{-1} \boldsymbol{d} - \boldsymbol{d}^T \Psi^{-1} W (I_M + W^T \Psi^{-1} W)^{-1} W^T \Psi^{-1} \boldsymbol{d}, \qquad (2)$$

where $\boldsymbol{d} = (\boldsymbol{x} - \boldsymbol{\mu})$ and $I_M$ is the $M \times M$ identity matrix. Note that $\Psi$ is diagonal and therefore its inverse and terms such as $\boldsymbol{d}^T\Psi^{-1}$ can be computed efficiently.

To address the $\log |C|$ term in equation 1 we make use of the identity $|A||D + EA^{-1}B| = |D||A + BD^{-1}E|$ to give

$$\log |WW^T + \Psi| = \sum_{i=1}^{D} \log \Psi_{ii} + \log |I_M + W^T \Psi^{-1} W|. \qquad (3)$$

Inverting a $D \times D$ matrix takes $O(D^3)$ time, as does the computation of its determinant. By using equations 2 and 3 determine the complexity of computing $\log p(\boldsymbol{x})$ in terms of $D$ and $M$.

## 4 Linear Dynamical System [35%]

1. We will now attempt to model the data using a linear Gaussian HMM, also known as a linear dynamical system (LDS). A LDS models a sequence of observations $\boldsymbol{x}_{1\ldots N}$

through a sequence of latent states $\boldsymbol{z}_{1\dots N}$ as follows (see §13.3 in Bishop, 2006):

$$\begin{aligned}
\boldsymbol{z}_n &= A\boldsymbol{z}_{n-1} + \boldsymbol{w}_n && (4)\\
\boldsymbol{x}_n &= \boldsymbol{\mu} + C\boldsymbol{z}_n + \boldsymbol{v}_n && (5)\\
\boldsymbol{z}_1 &= \boldsymbol{\mu}_0 + \boldsymbol{u} && (6)\\
\boldsymbol{w} &\sim \mathcal{N}(\mathbf{0}, \Gamma) && (7)\\
\boldsymbol{v} &\sim \mathcal{N}(\mathbf{0}, \Sigma) && (8)\\
\boldsymbol{u} &\sim \mathcal{N}(\mathbf{0}, V_0). && (9)
\end{aligned}$$

Below we will assume that $\Gamma$ and $\Sigma$ are diagonal matrices. $\boldsymbol{\mu}$ is a global offset, and $\boldsymbol{\mu}_0$ is the mean of the initial latent state distribution $p(\boldsymbol{z}_1)$.

Explain why a LDS might be appropriate for the data at hand. How does a LDS compare to FA?

2. Initialise the random number generator by calling `rand('seed', 0); randn('seed', 0)`. Fit a LDS with $M = 2$ dimensional latent state to the data using Prof. Ghahramani's function `lds`. Report the final likelihood of the data under the LDS model. Compare $C$ for the LDS and the $W$ obtained for FA with $M = 2$ using the `subspace` function. Comment on the results.

   HINT: `lds` returns a structure `net` that contains the parameters of the learned model and the log-likelihood. Note that `lds` uses different names for some of the parameters: The state noise covariance $\Gamma$ is `Q`, the observation noise covariance $\Sigma$ is `R` (this $D \times D$ dimensional diagonal matrix is returned as a $D \times 1$ dimensional vector), the initial state mean $\boldsymbol{\mu}_0$ is `x0` and the initial state covariance $V_0$ is `P0`.

3. For factor analysis we have used cross validation on the permuted data to select the appropriate number of latent factors for the data at hand. Would this be a sensible approach for determining the number of latent states of the LDS?

4. Use the provided function `ldspost` to infer the posterior distribution of the latent states. Plot the posterior means in 2D space using MATLAB's `line` function. How does the LDS projection compare to the PCA and FA projections?

5. From the inferred latent state you can reconstruct the original movie. Hand in code that generates the mean reconstruction (i.e. without adding observation noise). Store the reconstruction in a $581 \times 69$ matrix `sequence_Y_reconstructed`. Compare the reconstruction with the original movie, and comment on your results.

6. The LDS allows you to sample a new movie. Write code that samples a sequence of 581 frames. Store the sampled latent state in a $581 \times 2$ matrix `sequence_Z_sampled` and the sampled sequence in a $581 \times 69$ matrix `sequence_Y_sampled`. Sample the latent state both with and without noise, and sample the frames themselves without noise. Hand in the code, as well as a plot of the generated latent state sequences in 2D. Comment on the structure of the sampled latent state sequences and compare the two. Play the generated sequences using `skelPlayData`. Comment on your results.

   HINT: You may wish to use the Netlab function `gsamp`.

7. Compare the log likelihoods obtained for the dataset under the FA, GTM and LDS models. What does this tell us about the quality of the three as models for the data?

8. Consider a Hidden Markov model with a discrete state space, where each hidden state has an associated Gaussian emission distribution. How do you think this HMM would ideally fit to the data? Comment on both the Gaussian means and covariances, and the state transition matrix.

## 5 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.**
- Under the Resources heading on the PMR page there are a number of MATLAB tutorials listed including my "Introduction to MATLAB".
- You can find out more about most 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.
- `close all` closes all the figures. It helps if things get cluttered.
- Read about plots in the "Introduction to MATLAB" linked from the PMR homepage. Recall that the current figure can be save as an encapsulated postscript file `myplot.eps` using `print -deps myplot.eps`. I then find the unix command `epstopdf` helpful for creating `.pdf` files for inclusion in documents.