ABSTRACT

This thesis is comprised of two parts. The first part investigates whether a mutual information plot of wavelet packet transformed data can be used to determine the order in an Auto-Regressive (AR) process. The second part examines whether the mutual information of transformed electricity data can be estimated using a Gaussian Mixture of Models (GMM) where the transformation used is the wavelet pack transform.

Chapter 1

Introduction

1.1 Introduction to Load Forecasting

Load forecasting is an important tool in business for making key investment decisions. In the electricity supply industry long-term load forecasts can be used to plan the network whilst short-term forecasts could be used to purchase fuel or even spot possible power outages. In the telecommunications industry, decisions such as developing infrastructure (eg: extending broadband lines) and pricing of products/services would all be influenced by results from load forecasting. Government departments regularly forecast economic and demographic trends to assist decisions regarding budgeting, development of rail and road networks, and immigration policy.

Time series forecasting consists of creating a mathematical model based on past events and using this model to predict future points on the time series. The past events in which the model, $f(\cdot)$, is created from are known as the inputs, $U(k)$.

$$y(k) = f(U(k)) + \varepsilon(k) \tag{1.1}$$

Equation (1.1) is the typical equation for a model with the dependent time-series, $y(k)$, and where $\varepsilon(k)$ is a residual term. A model of this nature may be sufficient for multiple applications but for better results it is often necessary to transform the data prior to modelling [3].

$$A(y(k)) = f'(B(U(k))) + \varepsilon'(k) \tag{1.2}$$

Where $A(\cdot)$ and $B(\cdot)$ represents the input and output transform respectively, $f'(\cdot)$ denotes the new model and $\varepsilon'(k)$ is the new residual term. The aim of the transformations is to transform the input and dependent time-series into new domains, so as to increase the mutual information between the time series being modelled. This should lead to reduced errors and hence minimize some cost function (e.g. PMSE, Predictive Mean Squared Error).

Several types of transformations have been investigated such as Principal Component Analysis (PCA), Independent Component Analysis (ICA) and Fourier Transform (FT) [5]. However the Wavelet Packet Transform (WPT) has been shown [3] to be the preferred method as some time information is preserved in the transformed variables.

1.2 Time Series: Electricity Load Data

The time series' used to carry out this research was the Irish electricity load, Irish temperature and Irish humidity shown in figure 1.1 (a), (b) and (c) respectively. The three time series' were for the years between 1986 and 2000 inclusive and contained data at hourly intervals. From figure 1.1, one can see that there is a seasonal trend in the three time series'. Non-linear modelling techniques*[1] in general require stationary data for good results and hence all three time series were de-trended using a Kalman filter*[2]. The input/output (weather/load) data is then split into 24 parallel time series' each representing an hour of the day.

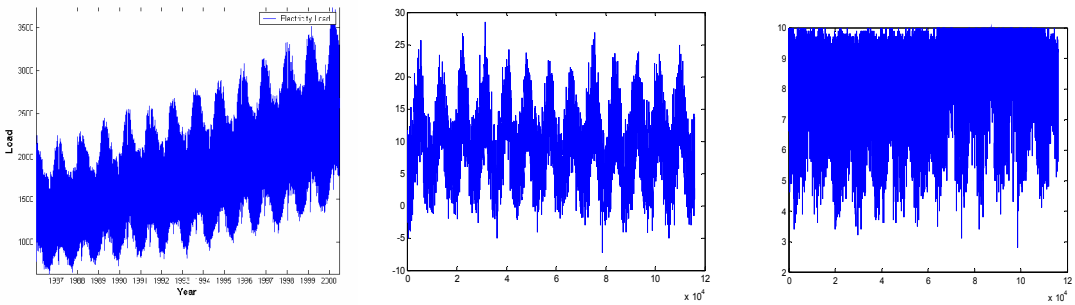

Figure 1.1: (a) Electricity Load Data (b) Temperature Date and (c)Humidity Data

1.3 Day Types

From just a quick inspection of the electricity load time series, certain assumptions can be made regarding the typical electricity load of a given day in the future. For example if the day was a winter working day, one could assume that its electricity demand would be greater than that of a summer weekend day. Since certain day's exhibit patterns that vary from others, it makes sense to break the calendar into day types and model each day type separately. Fay [4] discovered the day types in table 1.1 for Irish electricity load.

---

*[1] The non-linear modelling technique was a feed forward neural network. This neural network was developed by [Fay et al]. This report will not contain any detail regarding this modelling technique as it is irrelevant to the research carried out.

*[2] De-trending the data was carried out by Fay et al [3-5]. Therefore is wasn't part of my research and this report does not contain a detailed account of the de-trending process

No.	Day Type	Range
1	Early Winter Sundays	October-Christmas
2	Summer Sundays & Bank Holidays	April-September & Bank Holidays
3	Late Winter Sundays	January-March
4	Early Winter Working Days	October-Christmas
5	Summer Working Days	April-September
6	Late Winter Working Days	January-March
7	Early Winter Saturdays	October-Christmas
8	Summer Saturdays	April-September
9	Late Winter Saturdays	January-March
10	Christmas days	Christmas

Table 1.1: Day Types

The day-type modelled for the research documented in this report was day-type 6 (Late Winter Working Days).

1.4 Outline of Project

Much of the research carried out for this project is a continuation of the extensive work Fay [3-5] carried out in the area of Electricity demand forecasting. Hence the feed forward neural network used for modelling the data and the Wavelet Transfer Model was developed by Fay in [3-4].

The research conducted for this project comprises of two parts:

- Investigating whether the Wavelet Packet Transform can be used as an alternative to ACF/PACF for AR models: A 3-D plot of mutual information Vs input/output wavelet packet transformations for a randomly generated Auto-Regressive (AR) process looks non-random. Plots of any two randomly generated AR processes with the same lags are very similar. It has been suggested that there may be a link between these plots and the lag of an AR process. A detailed description of this part of my research can be found in Chapter 3.

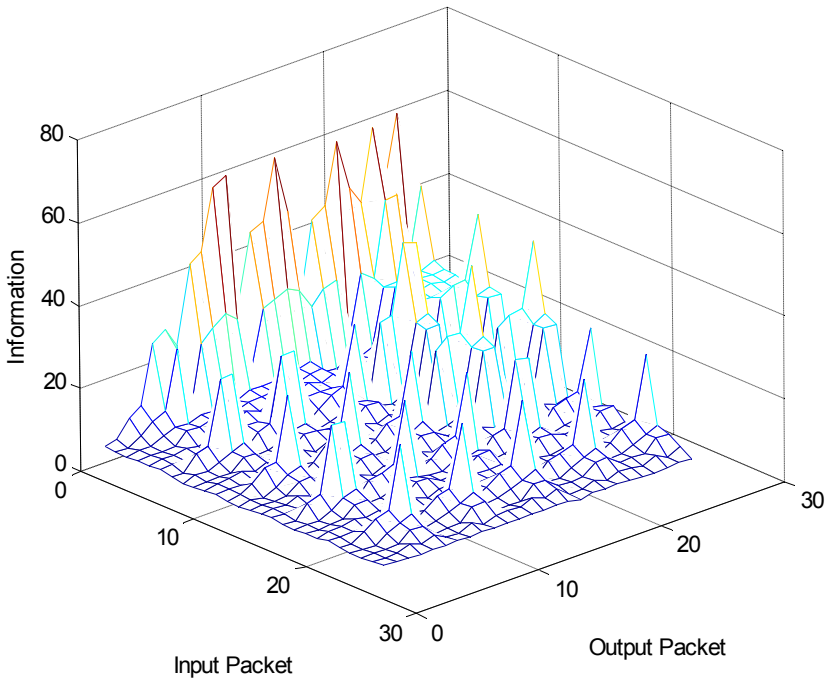

Figure 1.2: A plot of mutual information versus input/output packets

- Estimation of Mutual Information by fitting a Gaussian Mixture of Models: Mutual Information (MI) is an important metric in the study carried out by Fay in [3] in which it served the following two purposes:
    1. It is the basis behind choosing the optimal input/output wavelet packet transformations used to transform the time series prior to modelling.
    2. MI was used for input selection, inputs with a low MI were eliminated prior to modelling

  Since MI is such an important metric used in modelling, it is vital that mutual information is estimated accurately. In this study the input and output data are fitted using a Gaussian Mixture of Models (GMM). A weighted sum of the mutual information's contributed by every Gaussian kernel is then used as an estimation for the true mutual information of the data. A detailed description of Mutual Information, Wavelet Packet Transforms and Gaussian Mixture of Models can be found in sections 2.4, 2.2 and 4.2 respectively.

Chapter 2

Wavelets

2.1 Introduction to Wavelets

A wavelet is a mathematical function used to decompose a given function (or time series) into different frequency components. A wavelet transform is the representation of a function by wavelets and hence it transforms the time series into a time-frequency domain. A special case of the wavelet transform is the Fourier transform which transforms a time series into the frequency domain. The Fourier transform represents a time series by the sum of a series of sine functions. Specially designed pairs of orthogonal wavelets can be designed to decompose a time series into high and low frequency data. This is illustrated in figure 2.1, where the blue curve is the original time series (a noisy sine wave); the green and red curve is the result of convolving low and high pass filters with the original time series.

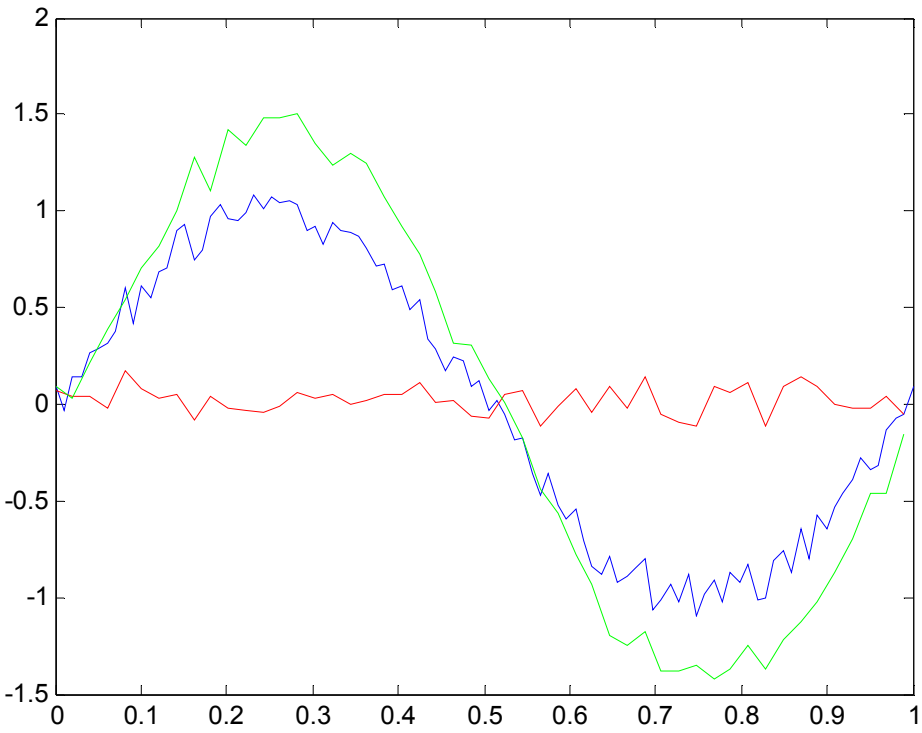

Figure 2.1: Blue: Original noisy sine wave; Green: Low frequency decomposition of the original wave; Red: High frequency decomposition of the original wave.

2.2 Wavelet Transfer Model

The Wavelet Packet Transform (WPT) decomposes a time-series into two separate time-series consisting of the high and low frequency components of the original data. The two remaining time-series' are then down-sampled by two*, leaving the same number of data-points as the original time-series. This process can be repeated a number of times leaving a tree-structure, such as the structures in Figure 2.2.

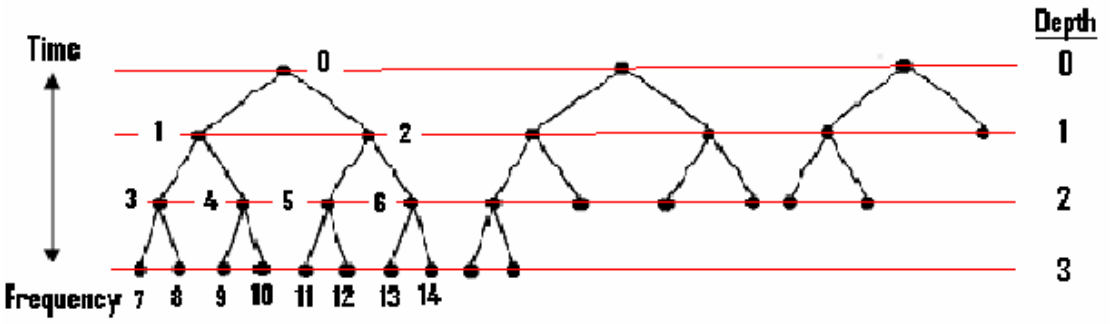

Figure 2.2: Diagrams of WPT tree structures. (a) the full Wavelet Packet tree to a depth of three, with the nodes numbered (b) Another wavelet packet with information in nodes {4,5,6,7,8} (c) information in nodes {2,3,4}

As can be seen from figure (2.2) the Wavelet Packet tree can take any structure as long as every parent node has both a low and high frequency child. The transformed time-series comprises of the coefficients belonging to all the nodes at the bottom of the tree. Every permutation of the tree structure to a chosen depth is used to transform both the input and the dependent time series. The transformations of the input and the dependent time-series that result in the largest mutual information are then chosen as the ideal transforms ($A(\cdot)$ and $B(\cdot)$) to model the data. The depth of the tree must be restricted for computational reasons. As the depth N increases the number of permutations explodes. This may be shown recursively using the following equation [3]:

$$s(n) = \left(s(n-1)+1\right)^2 \qquad n = 1,2,\ldots..,N \tag{2.1}$$

where $s(n)$ is the number of possible tree structures of depth N, $s(1) = 0$, and a depth of $N = 0$ indicates that no transform took place at all.

*↓2, every second data-point (co-efficient) of the filtered signal is deleted

2.3 Wavelet Basis Functions

Wavelet Bases, $(\psi_{i,j})$, are the small waves used to transform a discrete time series, $(y(t))$, by convolution to give *wavelet coefficients*, $(w_{i,j}^{y}(k))$. Therefore the transformed time-series can be represented as a function of the wavelet basis. For example, the basis function of the Fourier transform is $e^{i\omega}$, as every function in the time domain is a function of this basis. This transformation can be represented mathematically by the following equation:

$$w_{i,j}^{y}(k) = \int_{-\infty}^{+\infty} y(t)\psi_{i,j}^{*}(k)dt \tag{2.2}$$

where * denotes the complex conjugate of the wavelet. Two important operations of a wavelet transformation are *dilation* and *translation*. Dilation, a_0, stretches the wavelet along the time-frequency axis, (k), and translation, τ_0, shifts the wavelet along the time-frequency axis, (k). Dilation and translation are shown in equation (2.3).

$$\psi_{i,j}(k) = a_0^{i/2}\psi(a_0^i k - j\tau_0) \tag{2.3}$$

For discrete wavelet transformations the dilations, a_0, and translations, τ_0, are increased by fixed amounts whereas they are continuously increased for the continuous wavelet transform. A pair of orthogonal wavelets that decompose a time series into high frequency and low frequency data are known as wavelet and scaling functions respectively. A famous family of orthogonal wavelets are the Daubechies wavelets discovered by Ingrid Daubechies. There are 10 Daubechies wavelets each having a multiple of two discrete coefficients. The wavelet and scaling function pair used for this research were the Daubechies 4 wavelets and are shown in figure 2.3.

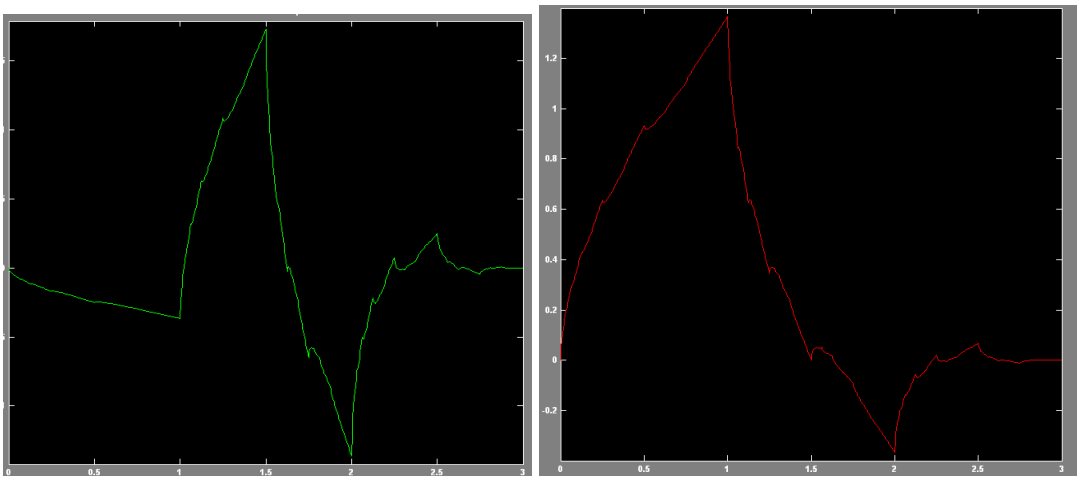

Figure 2.3: (a) Daubechies 4 Wavelet Function, (b) Daubechies 4 Scaling Function

2.4 Mutual Information

Mutual information is the measure of dependence between two variables. Therefore if two variables are independent the mutual information between them should be close to zero. Conversely, if two variables are a function of each other, then their mutual information should be large.

Mutual information can best be described by considering the entropy of the two dependent variables. The entropy of a variable is a measure of the uncertainty of the random variable and can be calculated using the following equation:

$$H(X) = \sum_{i}^{N} p(x_i).\log_n\left(\frac{1}{p(x_i)}\right) \tag{2.4}$$

Where $X = \{x_1, x_2, \ldots, x_N\}$, $p(x_i)$ is the probability of x_i occurring and n is the number of possible values in which x_i can hold. The following equation calculates the continuous (or differential) entropy, where $f_X(x)$ is the PDF (probability density function) of x.

$$H(x) = \int_{-\infty}^{\infty} f_X(x).\log\left(\frac{1}{f_X(x)}\right)dx \quad * \tag{2.5}$$

The conditional entropy of X given Y is calculated using the same formula only that the marginal probability in the equation is replaced by the conditional probability [9]. These two values of entropy can then be used to calculate the mutual information between two variables.

$$\begin{aligned} I(X,Y) &= H(X) - H(X \mid Y) \\ &= H(Y) - H(Y \mid X) = H(Y) - H(X,Y) + H(X) \end{aligned} \tag{2.6}$$

where $I(X,Y)$ is the mutual information between the two variables X and Y [8]. From equation xx it should become clear that mutual information measures the reduction in uncertainty in X which results from knowing Y. This relationship is illustrated in the Venn diagram of figure (2.4).

---

* The equation in the example is technically referred to as the differential entropy. Although this equation is analogous to the Shannon entropy, it has been proven that it is not the same thing [9]. It has been included in this report for explanatory reasons.

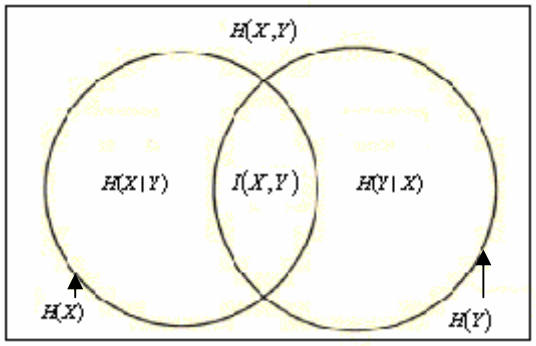

Figure 2.4: Relationship between the mutual information I(X;Y) and the entropies H(X) and H(Y)

From figure (2.4) one can observe the following two properties of mutual information

- Mutual Information is symmetric, i.e.

$$I(Y;X) = I(X;Y)$$

- The mutual information between X and Y is always non-negative, i.e.

$$I(X;Y) \geq 0$$

By subbing in all the necessary entropy equations into equation (2.6) the following expression for mutual information can be derived:

$$I(X;Y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_{X,Y}(x,y).\log\left(\frac{f_{X,Y}(x,y)}{f_X(x).f_Y(y)}\right).dxdy \qquad (2.7)$$

where $f_{X,Y}(x,y)$ is the joint PDF of the variables X and Y.

Estimating the distribution of $f_{X,Y}(x,y)$ is often not possible for multivariate continuous data [Fay]. By assuming that the data can be represented by Gaussian kernels, a reasonable approximation of the mutual information can be found using the following expression [3]:

$$I(X;Y) \approx \sum_{j=1}^{M}\int_X\int_Y \alpha j.Gj_{X,Y}(x,y).\log\left(\frac{Gj_{X,Y}(x,y)}{Gj_X(x)Gj_Y(y)}\right)dxdy\,,\ \sum_{j=1}^{M}\alpha_j = 1 \qquad (2.8)$$

where M is the number of Gaussian kernels that make up the data and α_j is known as the probability that a random data point belongs to kernel number j. $Gj_{U,Y}(u,y)$, $Gj_U(u)$, and $Gj_Y(y)$ are multivariate Gaussian distributions for the j^{th} kernel. Using the Expectation Maximization (EM) algorithm [2], estimates of the mean and covariance matrices for the

kernels may be found. Using these estimates the expression for the mutual information of two multivariate distributions reduces to:

$$I(X;Y) \approx \sum_{j=1}^{M} \alpha_j \cdot \left(-\frac{1}{2} \log \left[\frac{\left| \hat{C}j_{XY} \right|}{\left| \hat{C}j_X \right| \cdot \left| \hat{C}j_Y \right|} \right] \right) \tag{2.9}$$

where $|\cdot|$ denotes the determinant, $\hat{C}j_{XY}$, $\hat{C}j_X$ and $\hat{C}j_Y$ are the sample cross and auto-covariance matrices of the j^{th} kernel.

2.5 Input Selection

The purpose of input selection is to increase the *sampling density,* ρ, of the model by decreasing the number of dimensions in the input space. The sampling density of a model can be defined as the average number of samples in a unit hypercube of dimension m_0. For good function approximation the sampling density of the model must be relatively large. The following proportionality shows the relationship between the sampling density with the number of samples, *N*, and the dimension of the samples.

$$\rho \propto N^{1/m_0} \tag{2.10}$$

As can be seen from equation (2.10), the sampling density of a model increases if the dimension of the data is reduced. Therefore as the number of dimensions of the data increases, the number of sampling points necessary to model the data sufficiently increases at a much larger rate. This is often referred to as the *curse of dimensionality*. This is the reason why dimensions have to be eliminated, as there is not a large enough sample set available for the sampling density to remain at a sufficient level. The dimension of the weather data (input) used for this research is 72, representing a lag of three days (72 hours). Weather data has been shown to have an effect on electricity demand for just three days [3]. The sample size of the data was quite small (740) and hence it was decided to reduce the dimension to 7 [3]. This number is chosen subjectively with experience and depends on a number of factors such as the model used and the sample density required. The 7 dimensions that are retained as inputs are the dimensions that have the closest relationship to the output. Therefore the mutual information between every dimension of the input and the output are calculated. The 7 dimensions with the highest mutual information are the dimensions selected.

Chapter 3

Using the Wavelet Packet Transform as an Alternative to ACF/PACF

3.1 Auto-Regressive (AR) Models

An Auto-Regressive process is one of the most commonly used linear time-series models due to the ease of parameter estimation. An AR process is a model that forecasts future points on the time series using a function of previous points on the same time-series.

$$x(k) = \theta_1 x(k-1) + \theta_2 x(k-2) + \ldots\ldots \theta_N x(k-N) + \varepsilon(k) \tag{3.1}$$

The above process describes an AR(N) process where N is the lag, $[\theta_1, \ldots\ldots, \theta_N]$ are the parameters and $\varepsilon(k)$ is the error term. There are a number of ways to estimate the parameters of an AR model such as Least Square's and the Yule-Walker equations.

3.2 Auto-Correlation Function (ACF) & Partial Auto-Correlation Function (PACF)

The *auto-correlation* of a time series is the linear relationship of any point in a time series with previous points in the time series. Therefore the *auto-correlation function* is a plot of the auto-correlations of a time-series versus the lags at which the correlations are calculated. A proportion of the correlation of lags greater than two can be explained by its correlation with lower order lags. Therefore a better indication of the relationship between lags of a time-series would exclude the propagated correlation from lower order lags. This is the purpose of the *Partial Auto-Correlation Function* (PACF), which is similar to the ACF. The PACF displays the correlation between any point in a time-series and a lag of itself that cannot be contributed to the effects of correlation at lower order lags. Although both the ACF and PACF indicate where natural lags occur in an AR process, the PACF is a better method for determining the lags.

3.3 Can the WPT be Used as an Alternative to ACF/PACF

Figure 3.1 (a) displays a 3-D plot of mutual information versus the input and output packets. The input and output packets refer to different permutations of the Wavelet Packet tree structure shown in figure 3.1 (b). Both the input and output time series' were randomly

generated from an AR process of a known order. Several plots similar to the one in figure 3.1 (a) were generated, using inputs and outputs randomly generated from AR processes of the same order as the one's used in figure 3.1 (a). All of these plots contained spikes in the same places as the plot in figure 3.1 (a). The location of the spikes only changed when the order of the AR process generating the plot changed. Therefore it is reasonable to suggest that there may be a method of indicating the lags of an AR process by constructing plots similar to the one in figure 3.1 (a) and studying where the spikes occur.

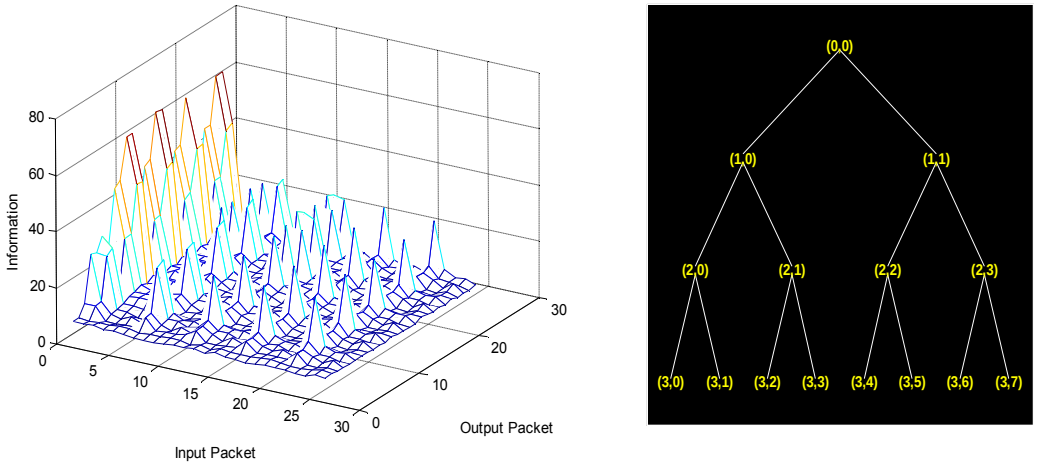

Figure 3.1: (a) Mutual Information plot and (b) A WPT tree

3.4 Generation of the Mutual Information Plot

In order to discover a method of indicating the lags of an AR process by using the plot in figure 3.1 (a), it is necessary to understand how the plot was generated. The following are the steps taken to generate the plot in figure 3.1 (a).

1. The first step involves the generation of the input and output time-series

   Input $\rightarrow x(k) = \theta . x(k-N)$ (3.1)

   Output $\rightarrow y(k) = \theta . x(k-N) + \varepsilon(k)$ (3.2)

   Where the error, $\varepsilon(k)$, and the first $(N-1)$ values of the time-series are randomly generated from a Gaussian distribution with a mean of zero and a variance, σ.

2. The Wavelet Packet Transform repeatedly decomposes the input time-series into high and low frequency data followed by a down-sampling by two. This results in fifteen different time series of varying lengths originating from the original time-series. The data points of the decomposed time-series are often referred to as the wavelet co-efficients. Each node of the tree in figure 3.1 (b) represents one of the fifteen time-

series'. The original time-series is represented by the root node of the tree and the length of each time-series is half the length of the time-series of its parent node.

3. Step 2 is repeated for the output time-series.
4. The data from each of the decomposed inputs and outputs are re-structured. Table's 3.1 & 3.2 show the way in which the data is restructured for the output and input respectively, where each box in each of the tables represents one of the fifteen time series'. N is the order of the AR process and n_1 is calculated using the number of points in the original time-series, n.

$$n_1 = \left\lfloor \frac{n}{N} \right\rfloor \tag{3.3}$$

The output data is structured in such a way that every N^{th} data point in the time series of the top line in table 3.1 belongs to the one vector.

$N \times n_1$							
$N/2 \times n_1$				$N/2 \times n_1$			
$N/4 \times n_1$		$N/4 \times n_1$		$N/4 \times n_1$		$N/4 \times n_1$	
$N/8 \times n_1$	$N/8 \times n_1$	$N/8 \times n_1$	$N/8 \times n_1$	$N/8 \times n_1$	$N/8 \times n_1$	$N/8 \times n_1$	$N/8 \times n_1$

Table 3.1: The size of the output data contained in every node of figure 3.1(b)

The input data contains a lag of N data points in each of its columns and hence there is twice as much input data as output data.

$2N \times n_1$							
$N \times n_1$				$N \times n_1$			
$N/2 \times n_1$		$N/2 \times n_1$		$N/2 \times n_1$		$N/2 \times n_1$	
$N/4 \times n_1$	$N/4 \times n_1$	$N/4 \times n_1$	$N/4 \times n_1$	$N/4 \times n_1$	$N/4 \times n_1$	$N/4 \times n_1$	$N/4 \times n_1$

Table 3.2: The size of the input data contained in every node of figure 3.1(b)

5. A nested iteration of the 26 output permutations of the WP tree with the 26 input permutations is performed. The input and output data for each one of these iterations is the combined data from each of the boxes in table 3.1 & 3.2 respectively that

represent the end nodes of the wavelet packet tree (e.g. the red nodes in figure 3.2). Therefore the size of the input and output data is $(2N \times n_1)$ and $(N \times n_1)$ respectively.

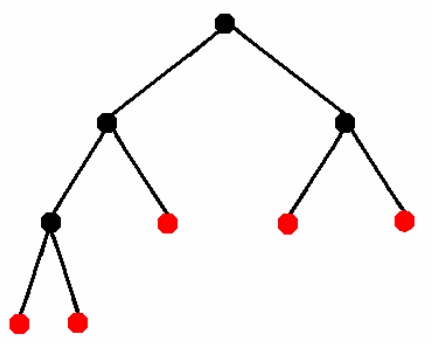

Figure 3.2: Tree structure number 18

6. The mutual information between the output and every vector of the input is estimated. The number of vectors in the input is then shrunk from *2N* vectors to a predefined value called the *shrinkage operator* [Fay et al]. The vectors retained were the ones that had the highest values of mutual information.
7. After the input is shrunk, an estimate of the mutual information can be calculated between the input and the output. A plot of all the estimates for every input-output combination should then result in a mutual information plot similar to the one in figure 3.1 (a).

3.5 Results – AR(8)

The procedure outlined in section 3.4 was carried out a number of times for the following input and output process:

$$\text{Input} \rightarrow x(k) = .2x(k-8) \tag{3.4}$$

$$\text{Output} \rightarrow y(k) = .2x(k-8) + \varepsilon(k) \tag{3.5}$$

The number of data points, *n*, in each time-series was 1000. Each time the mutual information was plotted, the 3-D graph contained spike's at the same locations as the plot in figure 3.3.

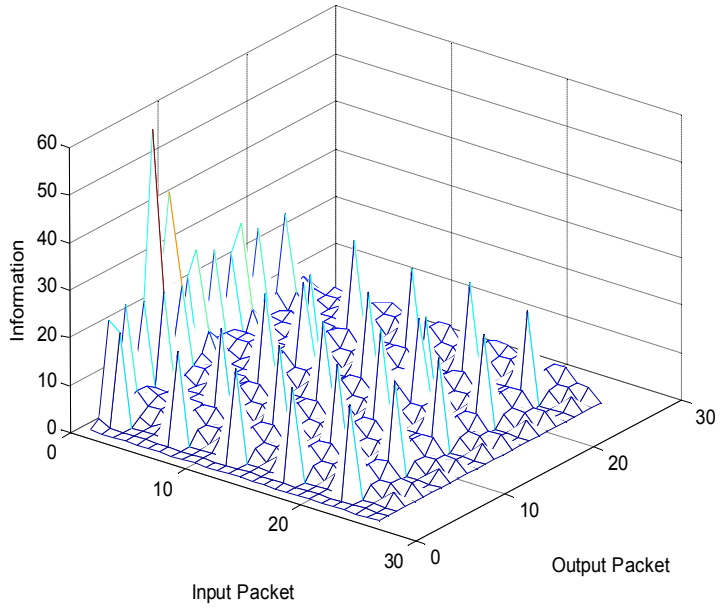

Figure 3.3: A mutual information plot of the AR(8) process

The WPT tree structure for the input and output in which spikes commonly occurred in for the AR(8) process of equation 3.5 are illustrated in figure 3.4 and 3.5 respectively.

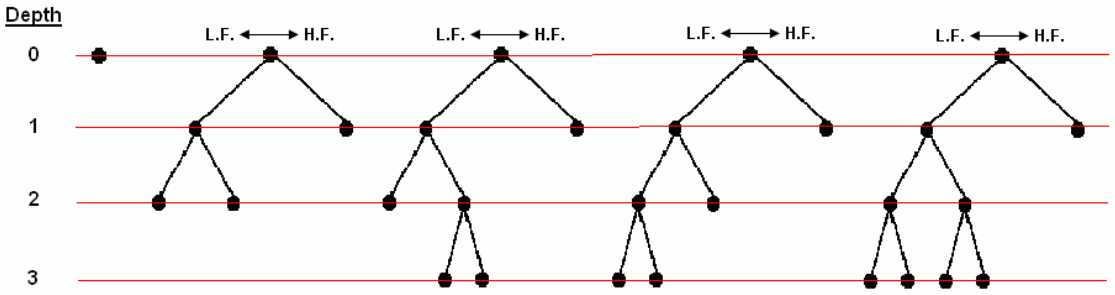

Figure 3.4: The input WPT tree structures in which a high level of mutual information was experienced. The packet numbers are from left to right: 1, 7, 12, 17 and 22. L.F. and H.F. stand for low and high frequency respectively.

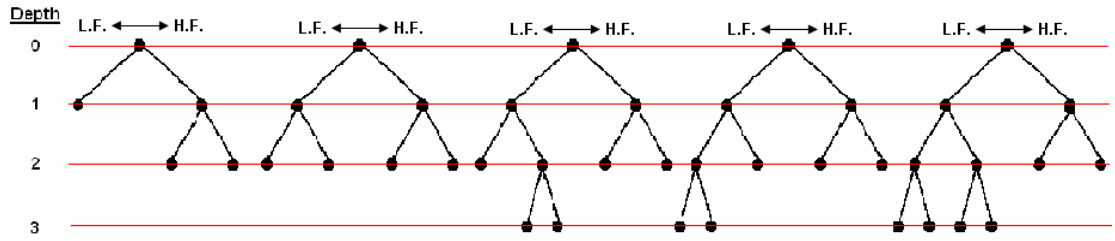

Figure 3.5: The output WPT tree structures in which a high level of mutual information was experienced. The packet numbers are from left to right: 3, 8, 13, 18 and 23.

From figure 3.4, it is clear that all of the tree structures, excluding packet number one, share a close resemblance. Each of these tree structures contain only one node on the high frequency side and hence up to half of the co-efficients belonging to each of these four trees are identical. In fact, three quarters of the co-efficients belonging to any one of these four trees are shared with two of the other tree structures in figure 3.4. Therefore it can be concluded that the mutual information estimates between the input co-efficients belonging to one of four packets mentioned and a given output are likely to be similar.

The tree structures in figure 3.5 also share a resemblance and hence they contain largely the same co-efficients. Therefore the mutual information estimates between the co-efficients belonging to the trees in figure 3.5 and a given input are likely to be similar. This conclusion coupled with the same conclusion for the inputs explains why the peaks in figure 3.5 occur in a grid like pattern. However, it does not explain why the peak mutual information occurs at these locations. In order to understand why the peaks are occurring in these locations it may be useful to examine which inputs are being eliminated and which are retained.

Since there were 1,000 data points, *N*, in the original time series, the size of the input matrix before shrinkage was (16×125). These 16 vectors were the combination of the vectors belonging to the boxes in table 3.2 that represented the input packet tree.

16×125							
8×125				**8×125**			
4×125		**4×125**		**4×125**		**4×125**	
2×125	**2×125**	**2×125**	**2×125**	**2×125**	**2×125**	**2×125**	**2×125**

Table 3.3: Table 3.2 for the AR(8) model

These 16 vectors were shrunk to 7 before the Mutual Information of that particular input/output combination was estimated. The following interesting observations were noted when examining the vectors that were obtained.

- In all of the cases examined, the majority of the retained vectors belonged to the low frequency side of the tree. The reason for this is that the high frequency components of a time-series are more random than the low frequency components and hence they represent noise. In nearly all cases either 5 or 6 of the 7 retained vectors were from the low frequency side of the tree.
- The vectors retained when the input packet number is one do not appear to have any pattern. The reason for this is that input packet number represents the time-series before it is decomposed and hence there are no low or high frequency divisions.
- All of the vectors retained for every input packet tree excluding packet number one are nearly always the same. However the ordering of the vectors with the highest mutual information does vary.

- Vector number 8 featured prominently in all of the retained matrices excluding the retained matrices associated with the input packet number one. This is an encouraging observation as the order of the generated process is 8, although the eighth (8^{th}) vector was not always the vector with the largest mutual information.
- The highest value of the mutual information between the input and output always occurred when the input packet number is one. The reason for this is that the input time series does not contain an error term (i.e. $x(k) = .2x(k-8)$), therefore the mutual information occurs when the input is not decomposed into low and high frequencies.

3.6 Varying the Shrinkage Operator

The number of input vectors shrunk has a big effect on the mutual information plot; since the input is reduced there is less data available to form a relationship with the output. A number of mutual information plots were generated. For every plot the shrinkage was varied and the plot was randomly generated using the following AR(24) process:

$$\text{Input} \rightarrow x(k) = .2x(k-24) \tag{3.6}$$

$$\text{Output} \rightarrow y(k) = .2x(k-24) + \varepsilon(k) \tag{3.7}$$

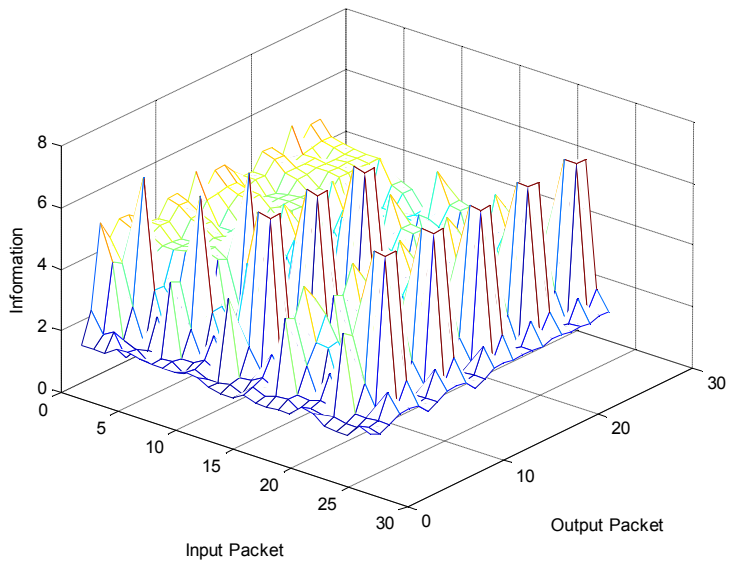

Figure 3.6: Mutual Information plot for the AR(24) process using a shrinkage operator of 2

The peaks in figure 3.6 are smaller and occur in more locations than in the other mutual information plots in figure 3.1 (a) and 3.3. Therefore the peaks are less distinct in this plot compared with the two previous plots. The reason why the mutual information of the peaks is

so low is because there are less co-efficients in the input to form a relationship with the output. The reason why there are so many peaks can be explained by the fact that most of the input tree structures are reduced to less complex tree structures. This concept is illustrated in figure 3.7, where the nodes in packet numbers 7 and 17 which contain co-efficients after the input is shrunk are highlighted in blue. This results in certain branches being effectively trimmed from both trees, leaving the same tree structure on the right hand side of figure 3.7. Since many of the tree structures are effectively the same after they are shrunk, there is less variation in the input packets. Therefore there is less variation in the mutual information plot of figure 3.7.

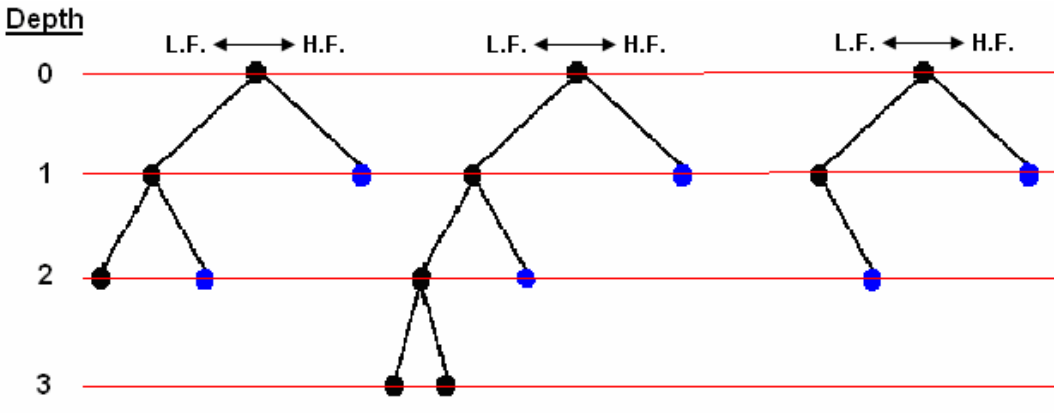

Figure 3.7: Wavelet Packet structures 7, 17 and the effective tree of both structures after input selection

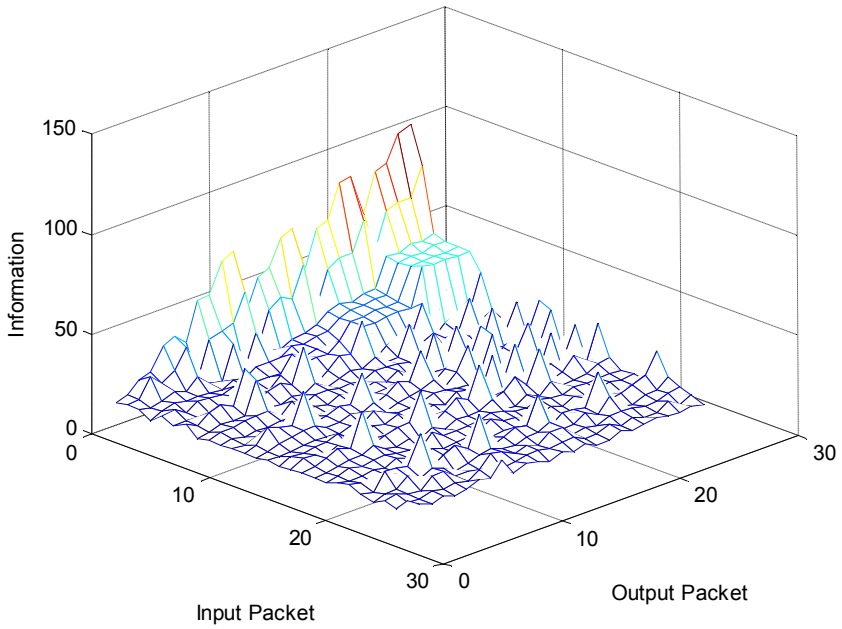

Figure 3.8: Information plot for the AR(24) process using a shrinkage operator of 16

The plot in figure 3.8 was generated from the same AR(24) process as figure 3.7 but the value of the shrink was 16. Note that this is an AR(24) process and hence the input before it is shrunk has 48 vectors. Using a value for shrink that is too high has the opposite problem to using a value that's too small. This problem is that the variation in the mutual information plot is too large whereas the variation was too small for low values of shrink. Since the value for the mutual information of the largest peaks in figure 3.8 is so large, some of the smaller peaks have become less distinct. Therefore best shrinkage operator to use for this purpose would be between 2 and 16. The plot in figure 3.9 was generated from the same process with a shrinkage operator of 8. The peaks in figure 3.9 are all very distinct and excluding peaks associated with input packet number 1 are all in arranged in a grid.

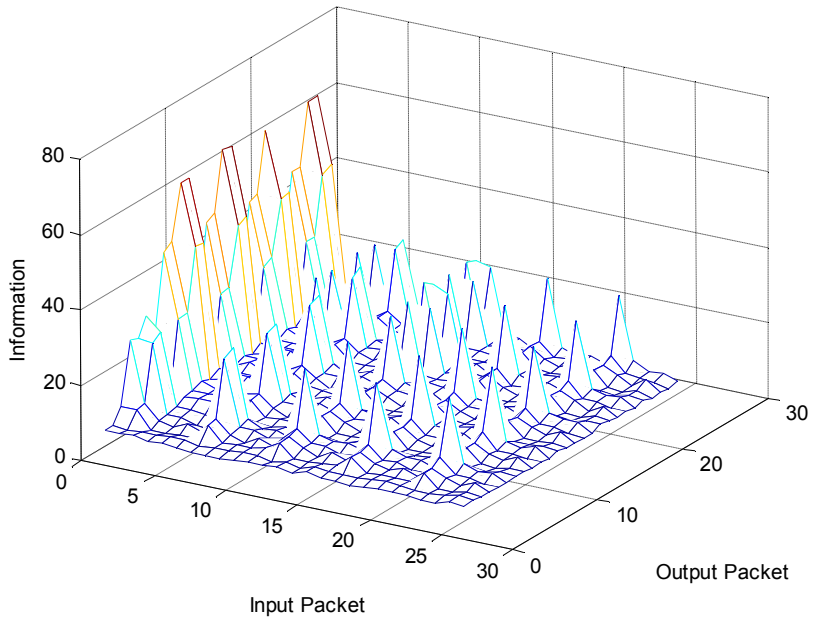

Figure 3.9: Information plot for the AR(24) process using a shrinkage operator of 8

3.7 Conclusions and Further Research

It is unlikely that a viable method of indicating the lag in an AR process can be discovered by using mutual information in conjunction with the wavelet packet transform. Even if a definite link between the lag of an AR process and the location of the spikes in the mutual information plot was proven, it is unlikely that it would be a worthwhile alternative to the PACF. The alternative method would be much more complicated and probably a lot less accurate at indicating the lags. The following is a list of further research that may help in discovering a definite link between the mutual information plots and the order of an AR process.

- This study has only examined results from two AR processes; i.e. AR(8) & AR(24). Although both AR processes led to similar results and conclusions, this may not be the case for AR processes of different orders.
- Examine the effects that varying the parameter, θ, has on the mutual information plots.
- Perform the same operation for *Moving Average* (MA) and *Auto-Regressive Moving Average* (ARMA) processes and investigate whether similar mutual information plots are generated. A moving average process is a process in which all points on a time series are functions of the error at previous points on the time series and a ARMA process is a combination of an AR and MA process.

Chapter 4

Estimation of Mutual Information

4.1 Introduction

In [3], Fay estimated the mutual information under the assumption that M=1. By making this assumption it is likely to have caused significant errors in the results. There are 676 input packet/output packet combinations for a wavelet packet tree to a depth of three. For each of these combinations in [3] the mutual information was estimated 73 times. The mutual information between each vector of the input and the whole output was computed for input selection. It was also estimated once to select the appropriate input/output transformations, resulting in 49,348 estimates for the whole procedure. It is reasonable to assume, that a number of these estimates would have been more accurate if the mutual information was computed using a weighted sum of the mutual information from separate Gaussian kernels. This can be achieved by using a Gaussian Mixture of Models.

4.2 Gaussian Mixture of Models (GMM)

A Gaussian mixture of models is a method of clustering data into groups. The Gaussian distribution is a very common distribution in statistics by virtue of the central limit theorem. This explains why the Gaussian mixture of models is such a popular clustering method and therefore it can be assumed that a lot of data sets could theoretically be generated using Gaussian distributions. Figure 4.1, illustrates a Gaussian mixture model containing two kernels with centres at (15,30) for the green kernel and (30,40) for the red kernel. Both kernels contain the same number of points (i.e. 500) but yet they have different shapes and sizes. The shape and size of the clusters are determined by the variance of every dimension within the cluster. The green cluster is roughly circular and hence the variances of both its directions are almost equal. The red cluster has a long elliptical shape and hence the variance of the data along the x-dimension must be much larger than the variance along the y-dimension. Often there can be considerable overlap between kernels in a Gaussian mixture model, such as the region in the top left of figure 4.1. When this happens it can be difficult to distinguish which kernel certain data points belong to and hence these data-points may be assigned to the wrong kernel. This will effect the estimation of the model parameters and hence it is an undesirable feature of Gaussian mixture models.

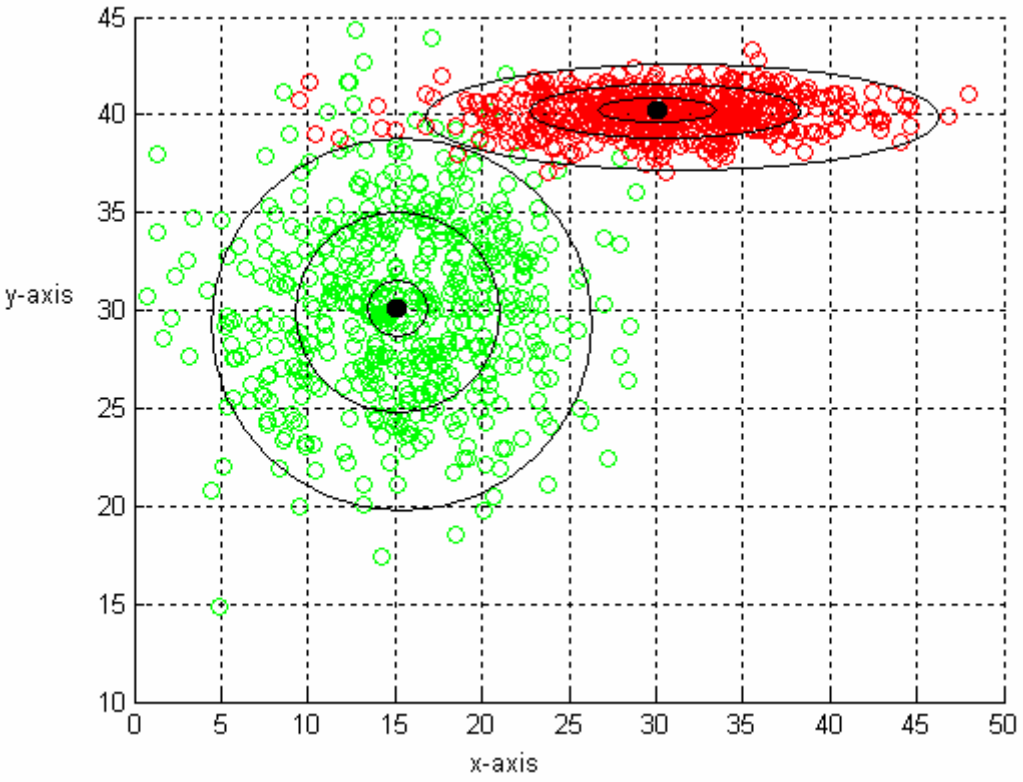

Figure 4.1: An example of a Gaussian Mixture of Models, the black dots represents the centres of both kernels.

4.3 EM Algorithm

The expectation-maximization (EM) algorithm is an iterative process for finding the maximum likelihood estimates of parameters in probabilistic models. The algorithm consists of two steps, an expectation (E) step followed by the maximisation (M) step. These two steps are repeated iteratively until the error value is below a certain predefined level or when the maximum number of iterations has been reached. The expectation step computes an expectation of the likelihood (J) by including the latent variables as if they were observed. The latent variables are parameters (θ) of the mixture model such as the means and covariance matrices of the mixture distributions. During the first iteration these parameters are given predefined estimates in order to calculate the likelihood, further iterations use the results from the maximization step to calculate the likelihood. The maximisation (M) step computes the maximum likelihood estimates of the parameters (θ) by maximising the expected likelihood found on the E step.

This process of alternating between estimating the unknowns and hidden variables is an old one but the first proof of convergence was demonstrated by Dempster, Laird and Rubin in 1977 [2].

The EM algorithm can be used to calculate the parameters of the Gaussian Mixture model in order to calculate the mutual information. These parameters are as follows, the prior probabilities (α_j), the centres (m_j), and the co-variance matrices (Σx_j). i.e.

$$\theta = \{\alpha_j, \vec{m}_j, \sum x_j\}_{j=1}^{k} \tag{4.1}$$

The prior of j is the probability of a randomly selected data point x_i belonging to kernel j. Hence the prior of j is the proportion of data points that belong to j and have the following properties.

$$\alpha_j > 0 \quad \text{and} \quad \sum_{j=1}^{k} \alpha_j = 1 \tag{4.2}$$

For the first iteration of the EM algorithm the priors of all the kernels were given the same estimates (i.e. 1 divided by the number of kernels). The centre ($\vec{m}_j$) is a vector who's length is the same as the number of dimensions in the data (M). Each entry in this vector is the mean of all the points associated with that kernel. The centres for the first iteration are initialised randomly using a Gaussian distribution with a zero mean and a variance of one. The co-variance matrices (Σx_j) are ($M \times M$) symmetric matrices each non-diagonal element represents the co-variance between different dimensions of the data and the diagonal elements represent the variance of each dimension. The co-variance matrices are all initialised to the identity matrix for the first iteration. Once all the latent variables are initialised the EM algorithm can compute the expectation of the likelihood (J or $P(j|\vec{x}_i)$) using the following equation [7]:

E step:

$$P(j \mid \vec{x}_i) = \frac{\alpha_i^{old} . G.(\vec{x}_i \mid j, \vec{m}_i^{old}, \sum x_i^{old})}{\sum_{i=1}^{k} \alpha_i^{old} . G.(\vec{x}_i \mid j, \vec{m}_i^{old}, \sum x_i^{old})} \tag{4.3}$$

M step:

$$\alpha_j^{new} = \frac{1}{N} \sum_{i=1}^{N} P(j \mid \vec{x}_i) \tag{4.4}$$

$$\vec{m}_j^{new} = \frac{\sum_{i=1}^{N} P(j \mid \vec{x}_i).\vec{x}_i}{\sum_{i=1}^{N} P(j \mid \vec{x}_i)} \tag{4.5}$$

$$\sum x_i^{new} = \frac{\sum_{i=1}^{N} P(j \mid \vec{x}_i).(\vec{x}_i - \vec{m}_i^{old}).(\vec{x}_i - \vec{m}_i^{old})^T}{\sum_{i=1}^{N} P(j \mid \vec{x}_i)} \tag{4.6}$$

To demonstrate the operation of the E-M algorithm synthetic data was generated. This synthetic data comprised of two Gaussian kernels with 500 data points each. The two kernels were generated using the following parameters:

Kernel 1 (Green)		**Kernel 2 (Red)**	
Prior Probability	0.5	Prior Probability	0.5
Mean (Centre)	(10,20)	Mean (Centre)	(30,40)
Covariance Matrix	$\begin{bmatrix} 14.63 & 1.06 \\ 1.06 & 14.58 \end{bmatrix}$	Covariance Matrix	$\begin{bmatrix} 17.37 & 0.06 \\ 0.06 & 17.1 \end{bmatrix}$

Table 4.1: GMM parameters of the two kernels in figure 4.1

Figures 4.2 (a) and 4.2 (b) illustrate where the centres were placed using the E-M algorithm iterated 50 times and 100 times respectively. The two blue dots in the figures corresponds to the random placement of the centres* prior to the first iteration. The two large black dots in the two figures represent the position of the estimated centres and the small black dots represent the real centres. In figure 4.2 (b) the large black dots cover the smaller dots and hence there were an adequate number of iterations to approximate the centres. This is not the case for figure 4.2 (a) where the modelled centres have not been updated enough. The two kernels estimated using 50 iterations differs greatly from the two kernels in which the data was generated from. After 50 iterations the EM algorithm identified the kernel circled in figure 4.2 (b) consisting of just 5 data points. All the other data points belonged to the other kernels and hence the priors were very inaccurate.

* The initial centres (blue dots) are placed in different positions on figure 4.2 (a) and figure 4.2 (b). The reason for this is because the two figures were generated by running the E-M algorithm twice using the same data. Hence two sets of random centres.

The co-variance matrices were also very inaccurate, the kernel with only 5 data points had very small variances, whereas the other kernel had large variances. The modelled parameters after 50 and 100 iterations can be found in Appendix A.

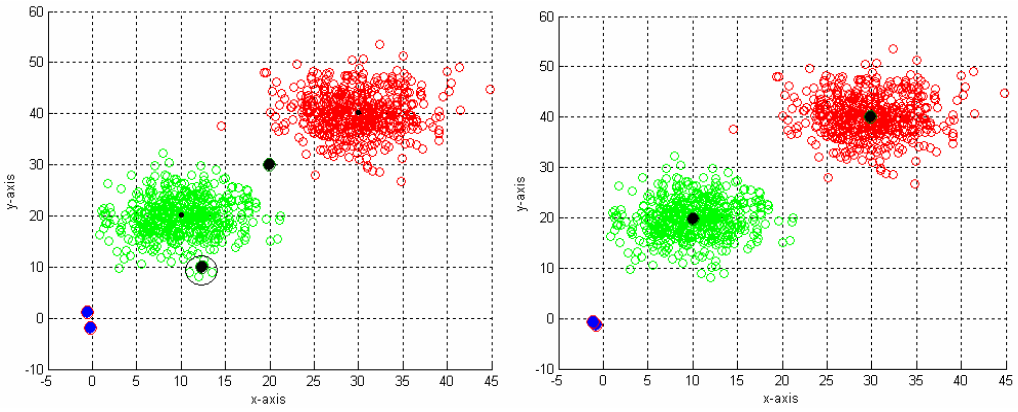

Figure 4.2: (a) The placement of the centres after 50 iterations of the EM algorithm (b) The placement of the centres after 100 iterations of the EM algorithm

4.4 How Many Clusters?

Selecting the optimum number of clusters is a well known problem in unsupervised learning. The problem remains an open problem in the literatures of pattern recognition and statistics although some heuristic techniques exist for solving this problem [10]. One such technique uses a special case of the Ying-Yang learning theory and system [7].

4.4.1 Bayesian Ying-Yang Learning System

The Bayesian Ying-Yang (BYY) learning system (also known as the Ying-Yang machine), was developed for pattern recognition purposes by Lei Xu in a series of papers between 1995 and 97 [10-12]. According to Xu [10], unsupervised and supervised learning problems can be summarized into the problem of estimating the joint probability distribution, $P(x, j)$. In Xu's example, X referred to the patterns in the input space whereas J referred to the representation space (i.e. the coded version of the input space). For a GMM application, X would refer to the training data points and J refers to a Gaussian kernel. Therefore x represents a particular component of X and j represents a component of J. Under the Bayesian framework [10], there are two representations for the distribution $P(x, j)$, the Ying and the Yang. These two representations are implemented using two models $M1$ & $M2$ called the Yang and the Ying respectively.

- $P_{M1}(x, j) = P_{M1}(j \mid x)P_{M1}(x)$: this performs the task of transferring a pattern into a code [10], analogous to this concept; it transfers input data points into Gaussian kernels. Therefore M_1 models the visible space X.

- $P_{M2}(x, j) = P_{M2}(x \mid j)P_{M2}(j)$: this performs the task of generating a pattern from a code [10], analogous to this concept; it performs the task of generating a set of data points from Gaussian kernels. Therefore M_2 models the invisible space J.

A Ying-Yang pair has four types of marital status depending on the minimisation or maximisation of its *Kullback divergences* [10]. A Kullback divergence is a measure of the difference between a true probability distribution, *P(X)*, and an arbitrary probability distribution, *P(J)*. The arbitrary distribution, *P(J)*, refers to a modelled distribution and hence this is an approximate distribution. This is the reason why the Kullback divergence is not symmetric, i.e. $K(M_1, M_2) \neq K(M_2, M_1)$. Equations for the Kullback divergences are shown in equations 4.7 and 4.8 below:

$$K(M_1, M_2) = \int P_{M1}(j \mid x)P_{M1}(x).\log\frac{P_{M1}(j \mid x)P_{M1}(x)}{P_{M2}(x \mid j)P_{M2}(y)}dxdy \qquad (4.7)$$

$$K(M_2, M_1) = \int P_{M2}(x \mid j)P_{M2}(j).\log\frac{P_{M1}(x \mid j)P_{M1}(j)}{P_{M2}(j \mid x)P_{M2}(x)}dxdy \qquad (4.8)$$

A combination of minimisation (chasing) and maximisation (escaping) on either of the divergences in equations (4.7) & (4.8) with respect to the models M_1 and M_2 will result in one of the four types of marital status [10]: marry, divorce, Ying chases & Yang escapes, and Yang chases & Ying escapes. The marital status type 'marry' refers to the minimization of the Kullback divergence with respect to both models M_1 and M_2, i.e. $min_{M1,M2}(K)$. This marital status is useful as it minimises the difference between the two model estimates (i.e. $P_{M1}(x,y)$ & $P_{M2}(x,y)$) of the joint distribution $P(x,y)$. It is also the basis behind the algorithm for choosing the optimum number of clusters detailed in section 4.4.2 and can be implemented using the alternative minimization (ALT-MIN) procedure. The ALT-MIN procedure is an iterative procedure with the following steps [11]:

Step 1: Fix $M_2 = M_2^{old}$, find $M_1^{new} = min_{M1}(K)$

Step 2: Fix $M_1 = M_1^{old}$, find $M_2^{new} = min_{M2}(K)$

This iterative procedure is guaranteed to converge [12].

4.4.2 Optimum number of Clusters Algorithm

This algorithm involves estimating a cost function *(J)* for every possible number of clusters and using this cost function to determine the optimum number of kernels. The cost function, $J(k, \theta)$, is derived by minimising the Kullback Divergence (equations (4.7) & (4.8)) using the ALT-MIN procedure mentioned in section 4.4.1. This results in the following expression [10]:

$$\min\left(KL\left(M_1, M_2\right)\right) = J\left(k, \theta\right) + c \tag{4.9}$$

where c is not a function of k and hence it is irrelevant. For a Gaussian mixture with the following probabilities:

- $P_{M1}(x) = \frac{1}{N}\sum_{i=1}^{N}\delta(x - x_i)$
- $P_{M2}(j) = \alpha_y$ (i.e. the probability)
- $P_{M1}(j|x) = P(j|x)$ (i.e. the probability calculated from the E-step of the EM algorithm)
- $P_{M2}(x|j)$ is generated from a Gaussian, $G(x, m_y, \Sigma_y)$

the following expression can be derived for the cost function, $J(k, \theta)$:

$$J(k,\theta) = J_1\left(k, \Sigma_j\right) + J_2\left(k, \alpha_j\right) + J_3\left(k, P\left(j \mid x\right)\right)$$

$$= \frac{1}{2}\sum_{j=1}^{k}\alpha_j.\log\left|\sum x_j\right| - \sum_{j=1}^{k}\alpha_j.\log\alpha_j + \frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{k}P\left(j \mid x_i\right).\log P\left(j \mid xi\right) \tag{4.10}$$

The steps in this algorithm are outlined below [7]&[10]:

Step 1: Set $k \leftarrow 1$, where k is the number of kernels

Step 2: Using the EM algorithm, equations (4.3)-(4.6), estimate the parameters of the Gaussian mixture model, $\theta = \{\alpha_j, \vec{m}_j, \sum x_j\}_{j=1}^{k}$, under the number of mixtures, k.

Step 3: Using the parameters found in step 2 calculate the cost function, $J(k, \theta)$, with the following Bayesian Ying-Yang criterion in equation (4.10).

Step 4: Increment the value for k, $k \leftarrow k+1$, repeat step 2 and 3 until k reaches a predefined maximum value.

$J(k)$ from equation (4.10) is a cost function and hence it measures the cost of choosing k as the optimal number of mixtures. Therefore the value of k that corresponds to the lowest $J(k)$ is deemed the optimum number of mixtures to model the data using a GMM. Although the above algorithm works well for a large set of data samples [6], the algorithm was occasionally unsuccessful in choosing the optimum number of kernels. This is particularly true when the data set is quite small. The performance of the above algorithm can be improved by incorporating the bootstrap technique with the Smoothing Expectation-Maximisation (SEM) algorithm.

4.4.3 Bootstrap Sampling

Bootstrapping is a re-sampling method for compiling statistics from a small initial sample. The process involves randomly selecting M samples from the initial sample of size N, where $N>M$. Bootstrap statistics such as the mean can then be calculated using the M samples. This process is repeated using re-sampling with replacement n times which results in n sets of bootstrap statistics. The average value of any bootstrap statistic (i.e. the average of the bootstrap means) can then be used as an estimate for the true statistic of the entire population. Since every bootstrap set contains a set of statistics, distributions and confidence intervals can be compiled for the various statistics.

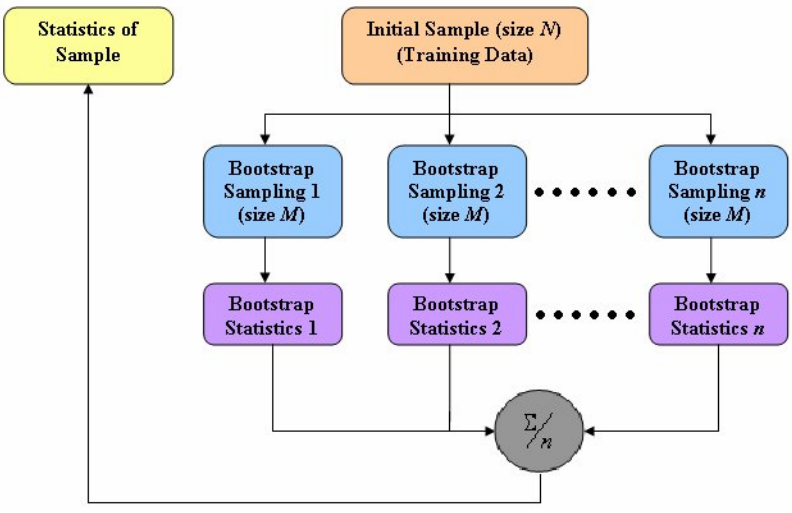

Figure 4.3: An illustration of the bootstrap technique

Bootstrapping was used in conjunction with the EM-algorithm to find the parameters of the Gaussian mixture model. Fifteen ($n=15$) bootstrap samples containing 100, (M=100), data points were randomly chosen from the initial sample size of N*. The EM algorithm was applied to each of the bootstrap samples resulting in 15 sets of GMM parameters. The average value for the prior, the centre and the co-variance matrix of every kernel across all the bootstrap samples were taken to give an estimate of the true parameters of the GMM. These parameters were used to calculate the posterior probability $P(j|x_i)$ and the cost function $J(k)$ was calculated for that particular value of k. Bootstrapping is an especially effective technique when used in conjunction with the EM algorithm. The reason for this is that the EM algorithm randomly assigns values taken from a Gaussian distribution to initialise the centres of each kernel before the first iteration. Therefore if the EM algorithm was applied to the same set of data twice the results would vary slightly. The Gaussian distribution from which the centres are randomly assigned from has a mean of zero and hence by using the bootstrapping technique the random effects are reduced. Bootstrap sampling is generally only used when the initial sample size, N, is relatively small. As the size of N increases, the effectiveness of bootstrap sampling decreases.

4.4.4 Smoothing Expectation-Maximisation (SEM)

To improve the results from the algorithm described in section 4.4.2 the Smoothing EM can be used instead of the EM algorithm. The SEM algorithm differs from the EM algorithm by employing covariance correction and hence eqn (4.6) is changed to the following eqn [6]:

$$\sum x_i^{new} = h^2 . I_d + \frac{\sum_{i=1}^{N} P(j \mid \vec{x}_i).(\vec{x}_i - \vec{m}_i^{old}).(\vec{x}_i - \vec{m}_i^{old})^T}{\sum_{i=1}^{N} P(j \mid \vec{x}_i)} \tag{4.11}$$

where I_d is a $d \times d$ dimensional identity matrix and h is the smoothing parameter.

* The size of N varied for all the tests carried out but its value was typically (400×k), where k is the number of kernels modelled.

In order to minimise the Kullback Divergence h should be estimated as [6]:

$$h = \arg\min J(h), \;\; J(h) = KL(k^*, \theta^*, h) \tag{4.12}$$

where k^* is the optimum number of kernels and θ^* is the optimum Gaussian parameters from equation (4.12) that minimises the Kullback Divergence. For fast implementation the *1/N* of the average distance approximation can be used as an estimate of the smoothing parameter, h [Guo].

$$h^2 = \frac{1}{dN^3} \sum_{i=1}^{N} \sum_{j=1}^{N} \left\| x_j - x_i \right\|^2 \tag{4.13}$$

Chapter 5

Results from GMM Analysis

5.1 Introduction

An expression for mutual information can be seen in equation 5.1, where $|\cdot|$ denotes the determinant, $\hat{C}j_{XY}$, $\hat{C}j_X$ and $\hat{C}j_Y$ are the sample cross and auto-covariance matrices of the j^{th} kernel [3]. Results obtained from this equation can differ by a large amount by choosing different values for M.

$$I(X;Y) \approx \sum_{j=1}^{M} \alpha_j \cdot \left(-\frac{1}{2} \log \left[\frac{\left|\hat{C}j_{XY}\right|}{\left|\hat{C}j_X\right| \cdot \left|\hat{C}j_Y\right|} \right] \right) \tag{5.1}$$

Two clearly distinct Gaussian kernels can be seen from figure 5.1. If the mutual information of the data from figure 5.1 was estimated using a value of M=1, then the mutual information would be over-estimated by a large amount. Whereas if the value was M=3 then the mutual information would probably be under-estimated. As the EM algorithm would probably try to fit a Gaussian kernel to outliers. This would reduce the variance and prior probability of one of the two kernels and hence reduce the mutual information.

5.2 Mutual Information Results using Synthetic Data

The example in figure 5.1, is merely used for illustration purposes and unfortunately Gaussian kernels found in real data are unlikely to be as well defined as the kernels in this example. A certain level of overlap between kernels would be expected when using these techniques to model real-life processes. With this in mind, synthetic data was randomly generated ensuring that there was some overlap, for each of the following models. The mutual information of the following mixture models were estimated using three different methods. The first method assumed that there was only one kernel present (i.e. M=1). The second method uses the correct number of kernels, along with the correct values for the prior probability and covariance matrices to estimate the mutual information by using equation (5.1). Hence the

second method is the best approximation for the mutual information and it is the value used to assess how good the other two methods are.

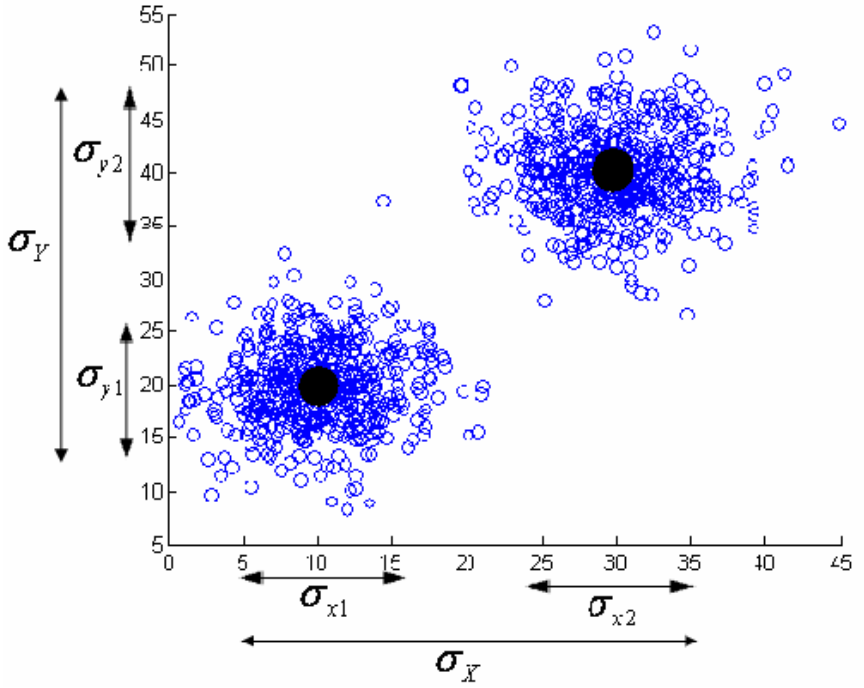

Figure 5.1: A scatter plot of two Gaussian kernels

The third method uses the correct number of kernels but uses the prior probability and covariance matrices obtained from the EM algorithm. All of the following models were simulated using the EM algorithm, iterated 2,000 times.

Two Kernels in Two Dimensions

Figure 5.2 displays 1,000 data points representing two Gaussian kernels. As can be seen from figure 5.2 there is considerable overlap between these the two kernels, despite the overlap the third method was much more accurate in estimating the mutual information than the first method.

Mutual Information Estimates

Method 1: 0.06355

Method 2: 0.00706

Method 3: 0.00557

In this example the parameters estimated by the EM algorithm were all very close to the actual parameters with the exception of the priors. The prior probabilities estimated by the

EM algorithm were 0.39 and 0.61 whereas the true priors were both 0.5. The reason why the priors were estimated incorrectly is due to the degree of overlap between the kernels. A comparison between the actual and modelled parameters for this GMM can be found in appendix B.

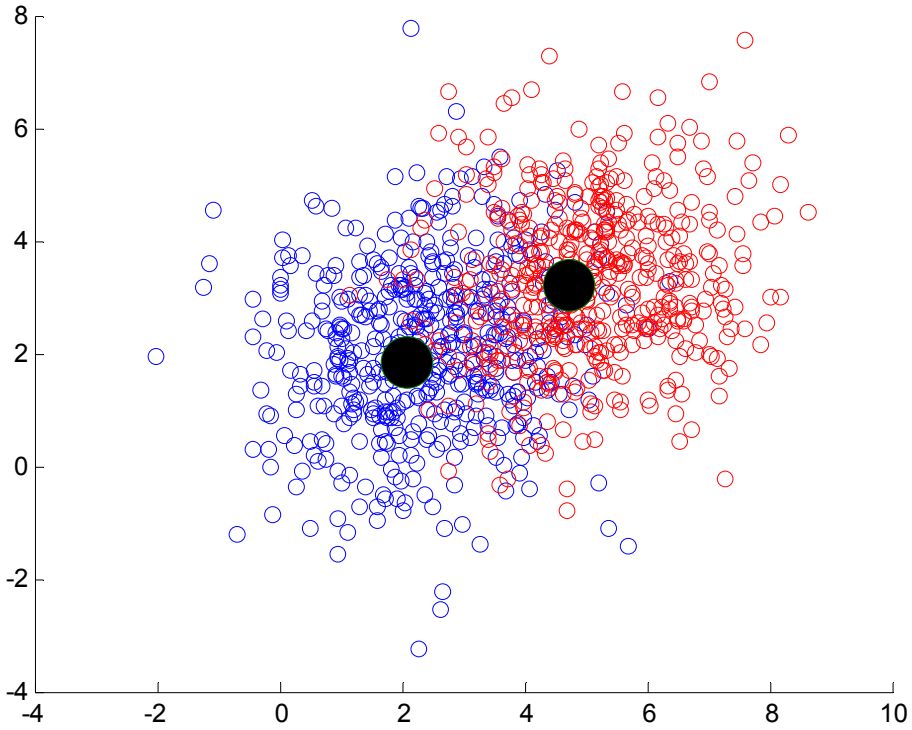

Figure 5.2: A scatter plot of two Gaussian kernels in two dimensions

Three Kernels in Two Dimensions

The GMM featured in figure 5.3 contains 1,500 data points representing three kernels. In this example the mutual information estimated using the third method was very close to the actual mutual information of the second method. As can be seen from figure 5.3, the variance of all the data is very large in comparison to the variance of each individual cluster.

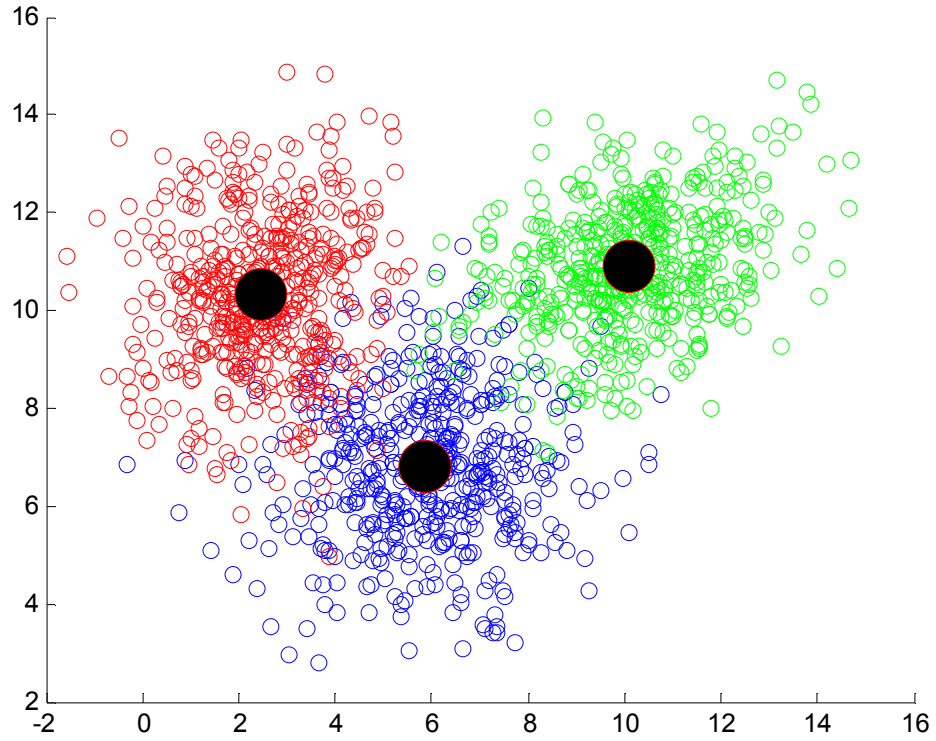

Figure 5.3: A scatter plot of three Gaussian kernels in two dimensions

Mutual Information Estimates

Method 1: 0.16244

Method 2: 0.02247

Method 3: 0.02393

The parameters estimated for this model using the EM algorithm were all good approximations of the actual parameters. A comparison between the actual and modelled parameters for this GMM can be found in appendix B.

Three Kernels in Three Dimensions

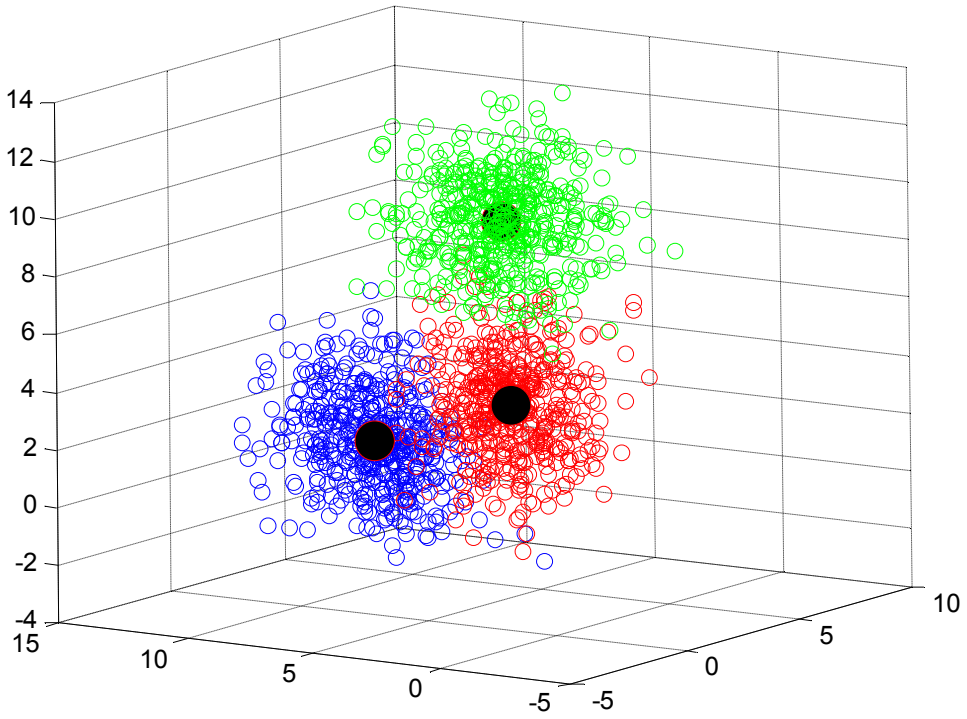

Figure 5.3: A scatter plot of three Gaussian kernels in three dimensions

The GMM featured in figure 5.3 contains 1,500 data points representing three kernels in three dimensions. In this example the mutual information estimated using the third method was very close to the actual mutual information of the second method. As can be seen from figure 5.3, the variance of all the data is very large in comparison to the variance of each individual cluster.

Mutual Information Estimates

Method 1: 0.20114

Method 2: 0.04710

Method 3: 0.03568

The parameters estimated for this model using the EM algorithm were all good approximations of the actual parameters. A comparison between the actual and modelled parameters for this GMM can be found in appendix B.

Three Kernels in Four Dimensions

The next model contained 1,500 data points representing three Gaussian kernels in four dimensions and hence it cannot be viewed graphically. The mutual information results were similar to all the previous examples. The third method was again a good approximation to the actual mutual information whilst the first method over-estimated.

Mutual Information Estimates

Method 1: 1.0774
Method 2: 0.1245
Method 3: 0.1306

The parameters estimated for this model were all very accurate and a comparison of the actual and modelled parameters for this GMM can be found in appendix B.

Three Kernels in Six Dimensions

The next model contained 1,500 data points representing three Gaussian kernels in six dimensions. In this example the first and third method over estimate the mutual information. By examining the estimated parameters in particular the priors, it becomes clear as to why the third method over estimates the mutual information. The prior probabilities that the EM algorithm estimated are the following: [0.3164 0.0006 0.6828] and the actual priors are: [.333 .333 .333]. Note that one of the priors suggests that the EM algorithm tried to fit a kernel around just one data point ($0.0006 \times 1{,}500 = .9 \approx 1$). It can also be concluded that the EM algorithm tried to fit one kernel around two whole kernels and some of the outliers from the other kernel. This would result in relatively large variances for the kernel which is weighted most heavily. Hence this would cause a high estimation for mutual information.

Mutual Information Estimates

Method 1: 0.6928
Method 2: 0.2952
Method 3: 0.5117

Only the parameters associated with one of the kernels were estimated accurately. These parameter estimates might be improved by increasing the number of iterations or perhaps the EM algorithm has found a local minima. If a local minima has occurred, the EM algorithm must be run again with different starting conditions. A comparison of the actual and modelled parameters for this GMM can be found in appendix B.

5.3 Results using the Optimum Number of Clusters Algorithm

In order to investigate the cluster number selection algorithm, fours sets of synthetic data were generated representing three and six Gaussian kernels in both two and three dimensions. The algorithm was performed twice for each data set, once using the Smoothing EM (SEM) algorithm and once using the ordinary EM algorithm. The smoothing parameter, *h*, was estimated using the formula (4.13) in section 4.4.4. The graph of the cost function versus the number of clusters is expected to look something like the graph in figure 5.4, [3].

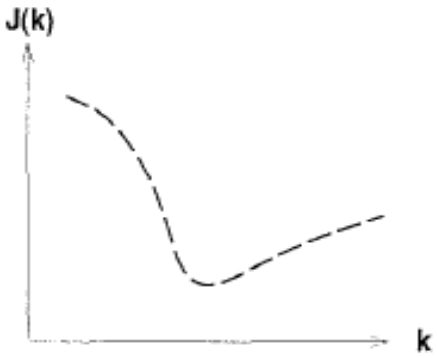

Figure 5.4: An ideal plot of J(k) versus k

Three Kernels in Two Dimensions

A scatter plot of the data contained in the first set can be seen in figure 5.5.

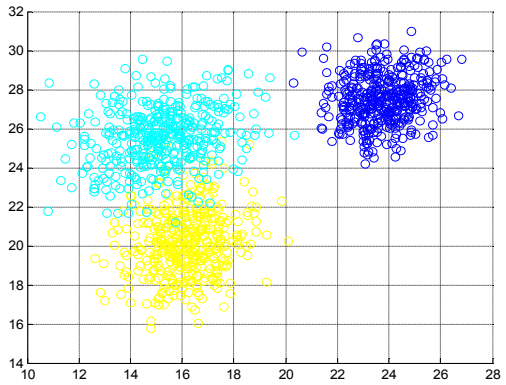

Figure 5.5: A scatter plot of three Gaussian kernels

Figure 5.6 (a) & (b) represents the results of the cost function *J(k)* versus *k* for the cluster number selection algorithm using the EM and SEM algorithms respectively. The global minimum in fig 5.6 (a) over-estimates the number of clusters, whereas the correct number of

clusters was selected in figure (b). Hence the smoothing parameter, *h*, in this example must have been estimated correctly.

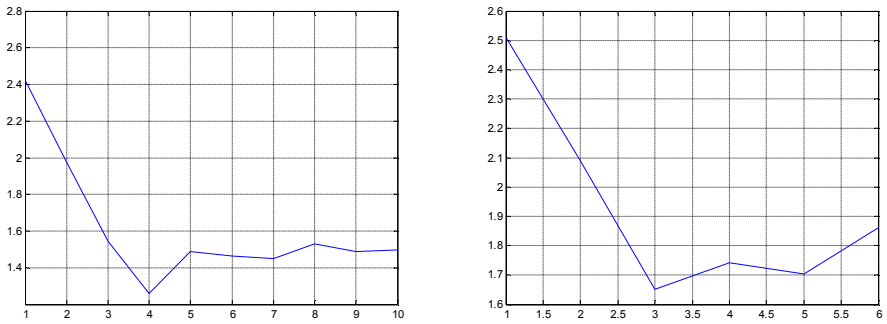

Figure 5.6: (a) Plot of J(k) versus k using the EM algorithm, (b) Plot of J(k) versus k using the SEM algorithm

Six Kernels in Two Dimensions

A scatter plot of the data contained in the first set can be seen in figure 5.7.

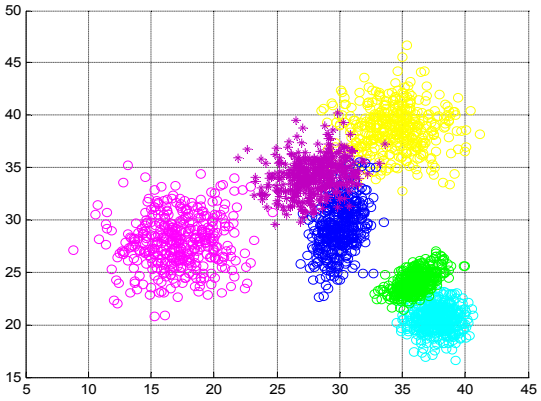

Figure 5.7: A scatter plot of six Gaussian kernels

Figure 5.8 (a) & (b) shows the results of the cost function *J(k)* versus *k* for the cluster number selection algorithm using the EM and SEM algorithms respectively. The global minimum in fig 5.8 (a) over-estimates the number of clusters. In figure 5.8 (b) there are local minimum's either side of the correct number clusters of 6. This would suggest that the SEM algorithm converged on a local minimum and estimated the parameters incorrectly for *k*=6. Therefore the smoothing parameter, *h*, in this example may have been estimated correctly.

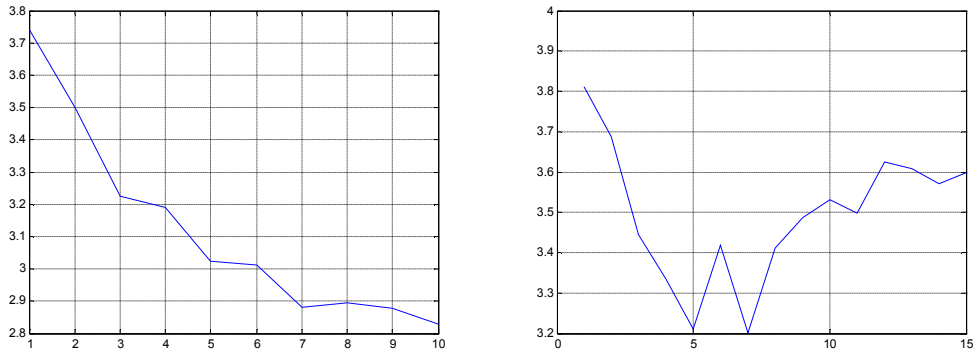

Figure 5.8: (a) Plot of J(k) versus k using the EM algorithm, (b) Plot of J(k) versus k using the SEM algorithm

Three Kernels in Three Dimensions

A scatter plot of the data contained in the first set can be seen in figure 5.9.

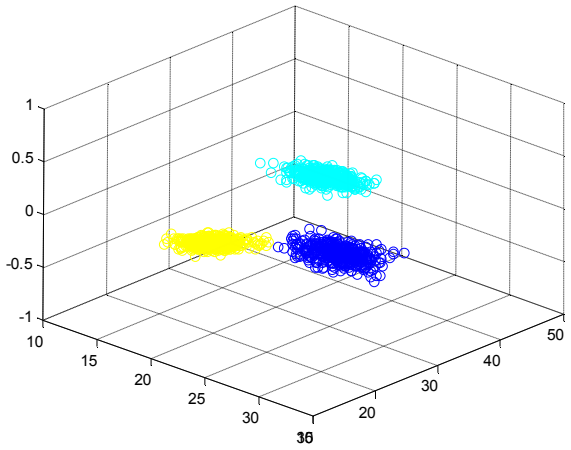

Figure 5.9: A scatter plot of six Gaussian kernels

The results in figure 5.10 (a) & (b) are very similar to the results for the first set of data. The smoothing parameter seems to have been estimated correctly and hence the number of clusters was correctly selected in figure 5.10 (b).

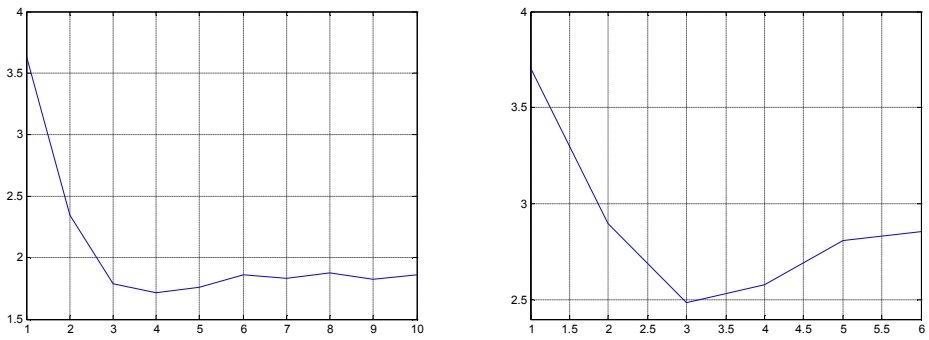

Figure 5.10: (a) Plot of J(k) versus k using the EM algorithm, (b) Plot of J(k) versus k using the SEM algorithm

Three Kernels in Three Dimensions

A scatter plot of the data contained in the first set can be seen in figure 5.11 (a)

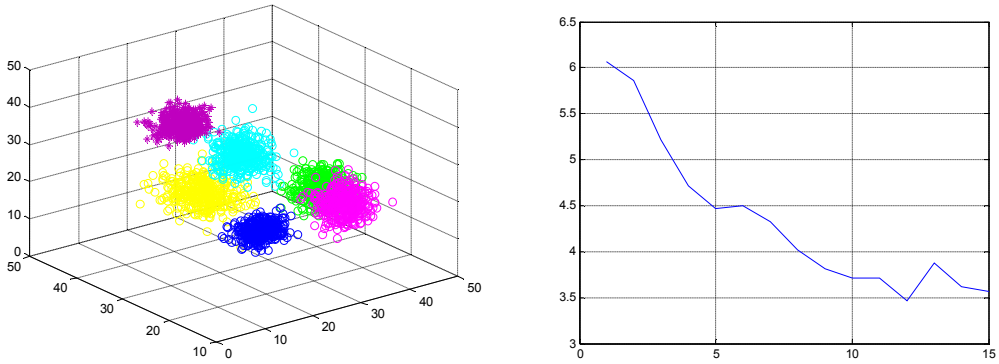

Figure 5.11: (a) A scatter plot of six Gaussian kernels in three dimensions, (b) Plot of J(k) versus k using the EM algorithm

Figure 5.11 (b) & 5.12 (a) shows the graphs of the cost function *J(k)* versus *k* using the EM and SEM algorithms respectively. In both cases the number of correct clusters was over-estimated with global minimums at 12 and 9. In figure 5.12 (b), the smoothing parameter, *h*, was increased from the value calculated using equation (4.13). This resulted in the correct value for the optimum number of clusters at *k*=6.

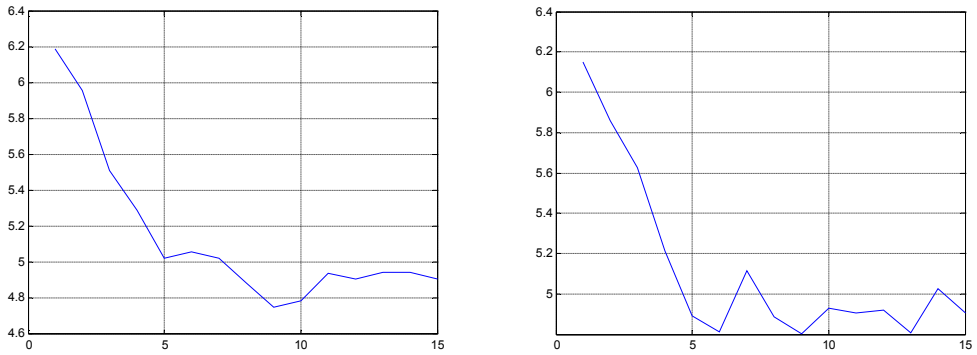

Figure 5.12: Plots of J(k) versus k using the SEM algorithm with (a) a smoothing parameter from equation (4.13) and (b) a smaller smoothing parameter

The experimental tests detailed above confirmed many of the conclusions in [6]. If *h*=0 or *h* is too small *k* is likely to be over-estimated. The converse is true if *h* is too large, k is underestimated [6].

5.4 Effects of Varying the Number of Dimensions on Mutual Information

The plot of figure 5.13 was created from randomly generated synthetic data. Five Gaussian kernels of five dimensions were generated randomly each having centres within a certain distance range of each other. The mutual information estimate of the first two dimensions of the first kernel was computed. This computation was repeated for dimensions 3 to 5. The same process was then repeated for the first two kernels using the third method of section 5.2. The process was then repeated for the first three kernels, and so on until the 20 mutual information estimates of figure 5.13 were computed. As illustrated in figure 5.13, the mutual information of each set of data (i.e. each number of kernels) increases with the number dimensions that were used to compute the mutual information.

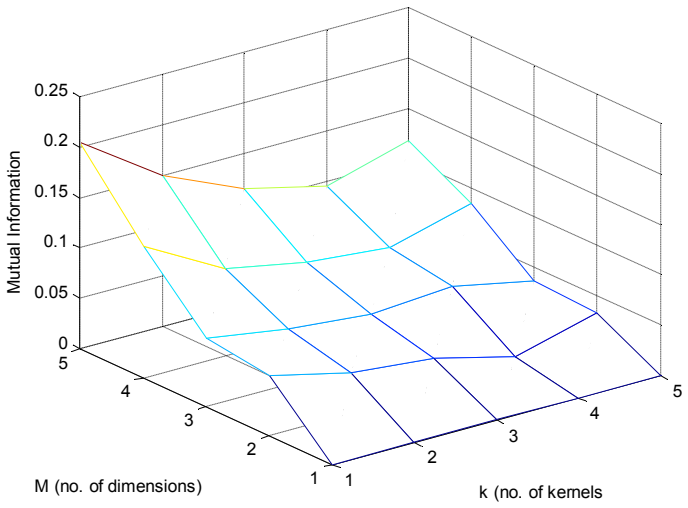

Figure 5.13: Mutual Information plot of synthetic data; both the dimension of the data and the number of kernels are both varied.

5.5 Difficulties Encountered using GMM and EM Algorithm

The following is a list of difficulties encountered when using Gaussian Mixture of Models in conjunction with the EM algorithm.

- *The optimum number of kernels are unknown:* From just a quick inspection of most of the plots examined in this study, one could guess the optimum number of kernels. If there was considerable overlap between the outliers of the two kernels (similar to the region at the top left of fig (4.1)) then it could be argued that this area should be modelled as a separate kernel. The task of choosing the optimum number of kernels

to model the data becomes more difficult when the dimension of the data is greater than three. This is because the data cannot be viewed graphically.

- *Too much overlap:* If there is too much overlap, the EM algorithm finds it difficult to distinguish which modelled kernel each data point belongs to. This can seriously distort the parameters, in particular the prior probabilities.
- *Too many dimensions:* As the number of dimensions increases the reliability of the EM algorithm decreases.
- *Uncertainty regarding the number of iterations necessary:* The choice of the number of iterations of the EM algorithm performed is a subjective one depending on experience. If the number of iterations is too low then the parameter estimation will be very inaccurate. This is the case in figure 5.14 where a larger number of iterations would have estimated the correct centres.

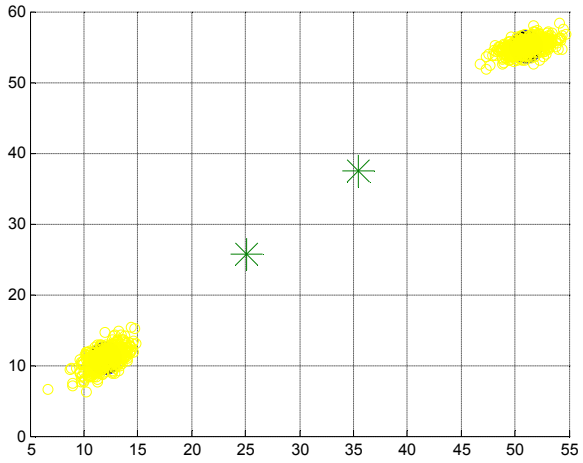

Figure 5.14: A scatter plot of two Gaussian kernels, the green asterisks marks the spots where the EM algorithm placed the centres

- *Local minima can be found instead of a global minimum* The EM algorithm has been proven to converge [1] to a local minimum, this however does not mean it will converge to the global minimum. Figure 5.15 illustrates a situation where the EM algorithm converged to a local minimum and hence an increased number of iterations are unlikely to improve parameter estimation. In figure 5.15, two of the estimated centres (top right & top left) are correct, one of the estimated centres (top left) fits a Gaussian kernel to a number of outliers and the other estimated centre (bottom left) fits two whole Gaussian kernels.

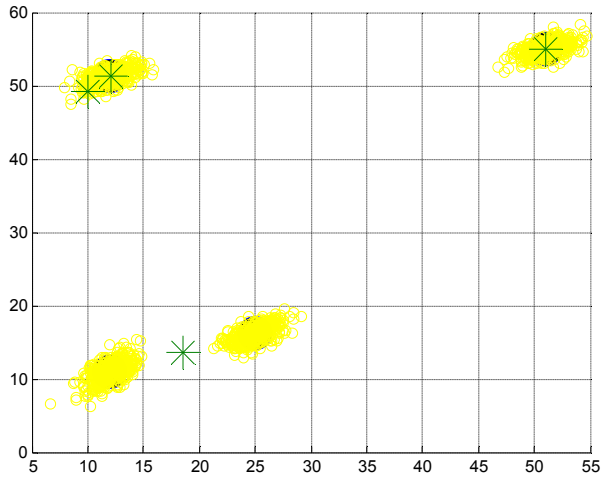

Figure 5.15: A scatter plot of four Gaussian kernels, the green asterisks marks the spots where the EM algorithm placed the centres

5.6 Conclusions

It has been shown in section 5.2 that the mutual information estimates can vary by a large amount if it is assumed that the data is represented by one kernel, as opposed to the optimum number of kernels. It has also been shown in section 5.3 that the 'number of clusters algorithm' can be used to select the optimum number of clusters with a considerable level of success. Therefore using a GMM in conjunction with the EM algorithm in order to estimate the mutual information does theoretically work. However this is not a trivial task as there are a number of problems which could make the simulations crash. The following is a list of the most common problems experienced which caused the simulations to crash.

- *Numbers are too small:* All simulation software languages such as MatLab contain a constant that represents the lowest positive number. If a certain variable that is calculated in the simulation results in a positive value below this constant, then the variable is assigned the value zero. If this variable is subsequently used as a denominator in a calculation further on in the simulation, then there is a 'divide by zero' problem which would cause the simulation to crash.
- *Numbers are too large:* All simulation software languages have a limit to the size of a number, any variable that exceeds this limit is deemed to be infinite. If the infinite variable is used in further calculations, the simulation will crash.
- *Singular Matrices:* A singular matrix refers to a matrix that is not of full rank and hence it cannot be inverted. The rank of a matrix is the number of rows (or columns) that are linearly independent. If the simulation requires a matrix to be inverted and if

this matrix is singular the simulation will crash. In such cases a pseudo-inverse can be found. A pseudo-inverse is process where every diagonal entry of a singular matrix is multiplied by a small unique number and the inverse of the resulting matrix is computed.

- *Cholesky Factorisation:* The Cholesky factorisation is a method of decomposing a positive-definite matrix, A, into a lower triangular matrix, L, and the transpose of L, an upper triangular matrix, L^*.

$$A = L.L^* \tag{5.2}$$

The Cholesky factorization is used in the EM algorithm to speed up computation when calculating the conditional probability of the data given that it is generated by a certain kernel, $P(X|J)$. The simulation will crash if A is not a symmetric, positive definite matrix. A positive definite matrix is a matrix in which all of its eigen-values are positive and symmetric means that all its entries are symmetric across its diagonal.

The mutual information is estimated 49,348 times in the procedure outlined in chapter two, with a tree depth of three. It has been shown that the EM algorithm should be used to estimate the parameters involved, therefore the 'optimum number of clusters' algorithm must be used. If this algorithm calculates the cost function up to a value of k=4 then the EM algorithm is run 197,392 times. From experience it can take the EM algorithm up to 2,000 iterations to adequately fit Gaussian kernels to low dimensional data. Therefore the whole procedure would consist of 394,784,000 iterations of the EM algorithm. This is a low estimate of the number of iterations. Ideally the number of clusters tested in the 'optimum number of clusters' algorithm would be more than four and 2,000 iterations is unlikely to be an adequate number of iterations to estimate the parameters of the high dimensional electricity data. Therefore the number of iterations would be much larger then 394,784,000. As a result it is likely that a variable at some stage will be ill conditioned and will cause the simulation to crash. The following refers to a list of methods that may prevent the simulation from crashing.

- *Normalise the Data:* Normalising the data refers to transforming all the data into the interval from -1 to 1 (or 0 to 1). The inverse of this transformation must then be applied to the results of the EM algorithm.
- *Modifying Problematic Variables:* This method requires identifying the variables within the code that are likely to cause the simulation to crash. These variables are then checked to see if they are ill-conditioned and modified if necessary. An example of this method is checking whether a matrix is of full rank before it is inverted. If the

matrix is not of full rank the pseudo-inverse is used instead. This method is not ideal as the results will be distorted; therefore it is important that the variables are modified as little as possible.

- *Principal Component Analysis (PCA):* There are 24 dimensions in the output electricity data; the mutual information is estimated between this data and the each of the dimensions of the input separately leaving 25 dimensions. The input dimensions associated with the 7 highest mutual information estimates are retained as the input. This leaves 31 dimensions in total for the final estimation between the input and output. This is a very high dimension which would be difficult to model and hence PCA could be used as a solution to this problem. PCA is a technique which can transform high dimensional data sets to lower dimensions. PCA transforms the data to a new co-ordinate system such that the greatest variance of any projection of the data lies on the first co-ordinate (the principal component), the second greatest variance on the second co-ordinate, and so on. The co-ordinates on the new co-ordinate system that have the smallest variance are eliminated, as these co-ordinates explain the least about the data.

The overall objective of this study was to find an enhanced method of estimating mutual information in order to improve the ability to forecast electricity demand. This objective has not been fulfilled due to the difficulties outlined above. Instead this study has shown that when the data is of a low dimension an improved estimate of mutual information can be obtained by using a Gaussian Mixture of Models. The shrunk input (weather) and output (electricity) data combined has 31 dimensions, this was found to be an unsuitable number of dimensions for the EM algorithm to accurately estimate the parameters of a GMM. It has been suggested that Principal Component Analysis may resolve the problems using high dimensional data and normalising the data may resolve other difficulties encountered.

References

[1] Dellaert, F., (2002). The Expectation Maximisation Algorithm, *College of Computing, Georgia Institute of Technology*

[2] Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B,* 39(1):1-38

[3] Fay, D. and Ringwood, J. (2005). A Wavelet Transfer Model for Tim-Series Forecasting. *National University of Ireland, Galway.*

[4] Fay D. 2004 *A Strategy for Short-Term Load Forecasting in Ireland*. PhD Thesis, Dublin City University.

[5] Fay, D., Ringwood, J.V., Condon, M., Kelly, M., (2003). 24-hour electrical load data – a sequential or partitioned time series? *Neurocomputing, 55: 469 – 498*

[6] Guo, P., Philip Chen, C.L. and R. Lyu, M., (2002). Cluster Number Selection for a Small Set of Samples Using Bayesian Ying-Yang Model. *IEEE Transactions on Neural Networks Vol. 13, NO. 3, May 2002, Pages 757-763*

[7] Li, X.Q., and King, I. (1999). Gaussian Mixture Distance for Information Retrieval. *IEEE Transactions on Neural Networks, 1999, Pages 2544-2549*

[8] Haykin, S. (1999). *Neural Networks, A comprehensive Foundation, Prentice Hall. Pages 492-497*

[9] MaCay, D.J.C. (2003). *Information Theory, Inference and Learning Algorithms. Pages 138-141*

[10] Xu, L. (1996). How Many Clusters? : A YING-YANG Machine Based Theory For A Classical Open Problem In Pattern Recognition. *IEEE Transactions on Neural Networks Vol 3. Pages 1546-1551*

[11] Xu, L. (1997). New Advances on Bayesian Ying-Yang Learning System With Kullback Separation Functionals. *IEEE Transactions on Neural Networks Vol 3. Pages 1942-1947*

[12] Xu, L. (1997). Bayesian Ying-Yang System and Theory as A Unified Statistical Learning Approach (II): From Unsupervised Learning to Supervised Learning and Temporal Modeling. *Theoretical Aspects of Neural Computation, Pages 25-42*

APPENDIX A

50 Iterations			
Kernel 1 (Green)		**Kernel 2 (Red)**	
Prior Probability	0.0051061	Prior Probability	0.99489
Mean (Centre)	(12.29,9.86)	Mean (Centre)	(19.97,30.04)
Covariance Matrix	$\begin{bmatrix} 0.56 & -0.005 \\ -0.005 & 1.73 \end{bmatrix}$	Covariance Matrix	$\begin{bmatrix} 114 & 101 \\ 101 & 118 \end{bmatrix}$

Table A.1: Estimated parameters after 50 iterations

100 Iterations			
Kernel 1 (Green)		**Kernel 2 (Red)**	
Prior Probability	0.50003	Prior Probability	0.49997
Mean (Centre)	(10.06,19.77)	Mean (Centre)	(29.82,40.11)
Covariance Matrix	$\begin{bmatrix} 14.6 & 1.07 \\ 1.07 & 14.56 \end{bmatrix}$	Covariance Matrix	$\begin{bmatrix} 17.33 & 0.06 \\ 0.06 & 17.06 \end{bmatrix}$

Table A.2: Estimated parameters after 100 iterations

APPENDIX B

Actual		Simulated	
Priors	[0.5 0.5]	Priors	[0.39 0.61]
Centres	[2.3740, 2.0938] [5.0580, 3.2785]	Centres	[2.0648, 1.8398] [4.9871, 3.2044]
Covariance Matrices	$\begin{bmatrix} 1.8349 & 0.2432 \\ 0.2432 & 2.1855 \end{bmatrix}$ $\begin{bmatrix} 1.6967 & 0.1896 \\ 0.1896 & 1.9265 \end{bmatrix}$	Covariance Matrices	$\begin{bmatrix} 2.0777 & 0.2462 \\ 0.2462 & 1.9116 \end{bmatrix}$ $\begin{bmatrix} 1.5934 & 0.0067 \\ 0.0067 & 2.0501 \end{bmatrix}$
M.I.	0.007069	M.I.	0.00477

Table B.1: EM Estimated and actual parameters off two Gaussian kernels in two dimensions.

Actual		Simulated	
Priors	[.333 .333 .333]	Priors	[0.33 0.328 0.332]
Centres	[5.8777 6.8118] [2.4607 10.2814] [10.0818 10.8931]	Centres	[5.850, 6.824] [2.451, 10.32] [10.07, 10.89]
Covariance Matrices	$\begin{bmatrix} 2.619 & 0.1416 \\ 0.1416 & 2.3365 \end{bmatrix}$ $\begin{bmatrix} 1.6394 & 0.1843 \\ 0.1843 & 2.4865 \end{bmatrix}$ $\begin{bmatrix} 2.5047 & 0.6505 \\ 0.6505 & 1.5978 \end{bmatrix}$	Covariance Matrices	$\begin{bmatrix} 2.6863 & 0.0959 \\ 0.0959 & 2.3308 \end{bmatrix}$ $\begin{bmatrix} 1.7248 & 0.2565 \\ 0.2565 & 2.3718 \end{bmatrix}$ $\begin{bmatrix} 2.5529 & 0.6682 \\ 0.6682 & 1.5767 \end{bmatrix}$
M.I.	0.02393	M.I.	0.02247

Table B.2: EM Estimated and actual parameters off two Gaussian kernels in three dimensions.

Actual		Simulated	
Priors	[.333 .333 .333]	Priors	[0.33 0.33436 0.33563]
Centres	[1.5232, 8.4338, 1.5850] [2.2757, 3.7415, 3.0924] [3.7616, 5.3070, 9.0070]	Centres	[1.5217, 8.4027, 1.5789] [2.2733, 3.7157, 3.1102] [3.7661, 5.3266, 8.9878]
Covariance Matrices	$\begin{bmatrix} 1.736 & 0.206 & 0.139 \\ 0.206 & 2.362 & 0.927 \\ 0.139 & 0.927 & 2.2634 \end{bmatrix}$ $\begin{bmatrix} 1.733 & 0.432 & 0.609 \\ 0.432 & 2.280 & 0.697 \\ 0.609 & 0.697 & 2.447 \end{bmatrix}$ $\begin{bmatrix} 2.483 & 0.493 & 0.660 \\ 0.493 & 3.017 & 0.792 \\ 0.660 & 0.792 & 1.838 \end{bmatrix}$	Covariance Matrices	$\begin{bmatrix} 1.744 & 0.214 & 0.151 \\ 0.214 & 2.362 & 0.884 \\ 0.151 & 0.884 & 2.183 \end{bmatrix}$ $\begin{bmatrix} 1.674 & 0.416 & 0.558 \\ 0.416 & 2.274 & 0.693 \\ 0.558 & 0.693 & 2.464 \end{bmatrix}$ $\begin{bmatrix} 2.494 & 0.465 & 0.660 \\ 0.465 & 3.017 & 0.792 \\ 0.660 & 0.792 & 1.838 \end{bmatrix}$
M.I.	.0471	M.I.	0.03568

Table B.3: EM Estimated and actual parameters off three Gaussian kernels in three dimensions.

Actual		Simulated	
Priors	[.333 .333 .333]	Priors	[0.335 0.330 0.334]
Centres	[2.256, 6.821, 5.306, 8.328] [4.637, 8.314, 6.931, 3.091] [8.311, 3.797, 2.400, 1.552]	Centres	[2.267, 6.817, 5.311, 8.305] [4.637, 8.334, 6.945, 3.082] [8.306, 3.805, 2.404, 1.553]
Covariance Matrices	2.365 0.250 0.324 0.373 0.250 1.919 0.688 0.719 0.324 0.688 1.968 0.366 0.373 0.719 0.366 2.218 1.942 0.095 0.196 0.529 0.095 2.000 0.661 0.301 0.196 0.661 2.061 0.586 0.529 0.301 0.586 2.343 2.179 0.851 0.770 0.081 0.851 1.707 0.592 0.639 0.770 0.592 1.857 0.141 0.081 0.639 0.141 1.988	Covariance Matrices	2.374 0.260 0.325 0.338 0.260 1.919 0.664 0.734 0.325 0.664 1.960 0.343 0.338 0.734 0.343 2.276 1.931 0.071 0.214 0.544 0.071 1.960 0.654 0.315 0.214 0.654 2.043 0.594 0.544 0.315 0.594 2.335 2.178 0.834 0.756 0.081 0.834 1.720 0.601 0.6447 0.756 0.601 1.855 0.1423 0.081 0.644 0.142 1.9732
M.I.	0.12446	M.I.	0.1306

Table B.4: EM Estimated and actual parameters off three Gaussian kernels in three dimensions.

Actual		**Simulated**	
Priors	[.333 .333 .333]	Priors	[0.3164 0.0006 0.6828]
Centres	[4.5, 1.6, 3.8, 2.3, 1.4, 3.8] [9.1, 2.2, 3.1, 1.5, 6.7, 5.4] [3.0, 8.2, 6.8, 3.7, 3.8, 3.1]	Centres	[4.4, 1.5, 3.8, 2.3, 1.4, 3.8] [-1.0, 7.3, 1.3, 1.6, 3.5, -2.2] [6.1, 5.2, 5.0, 2.6, 5.2, 4.2]
Covariance Matrices	2.1 0.8 0.1 0.8 0.9 0.6 0.8 2.5 0.8 0.5 0.3 0.3 0.1 0.8 1.9 0.3 0.0 0.2 0.8 0.5 0.3 2.3 0.8 0.2 0.9 0.3 0.0 0.8 1.7 0.8 0.6 0.3 0.2 0.2 0.8 1.8 1.8 0.0 0.4 0.1 0.4 0.0 0.0 1.5 0.2 0.8 0.1 0.5 0.4 0.2 2.6 0.7 0.9 0.0 0.1 0.8 0.7 2.5 0.6 0.3 0.4 0.1 0.9 0.6 1.8 0.2 0.0 0.5 0.0 0.3 0.2 2.4 2.6 0.4 0.7 0.7 0.7 0.5 0.4 1.9 0.4 0.4 0.8 0.2 0.7 0.4 2.5 1.1 0.6 1.2 0.7 0.4 1.1 2.4 0.5 0.7 0.7 0.8 0.6 0.5 2.7 1.1 0.5 0.2 1.2 0.7 1.1 2.4	Covariance Matrices	2.0 0.7 0.0 0.8 0.8 0.6 0.7 2.3 0.8 0.5 0.2 0.2 0.0 0.8 1.9 0.3 0.0 0.2 0.8 0.5 0.3 2.3 0.7 0.2 0.8 0.2 0.0 0.7 1.6 0.8 0.6 0.2 0.2 0.2 0.8 1.8 0.1 0.0 0.3 0.1 0.0 0.3 0.0 0.2 -0.0 -0.1 0.1 -0.0 0.3 -0.0 0.8 0.4 0.0 0.9 0.1 -0.1 0.4 0.5 -0.0 0.4 0.0 0.1 0.0 -0.0 0.1 0.0 0.3 -0.0 0.9 0.4 0.0 1.1 11.1 -8.5 -4.8 -2.7 4.8 3.6 -8.5 10.4 5.7 3.7 -3.6 -2.9 -4.8 5.7 5.8 2.8 -1.7 -1.4 -2.7 3.7 2.8 3.6 -0.9 -0.6 4.8 -3.6 -1.7 -0.9 4.4 2.3 3.6 -2.9 -1.4 -0.6 2.3 3.6
M.I.	0.2952	M.I.	0. 5117

Table B.5: EM Estimated and actual parameters off two Gaussian kernels in six dimensions.