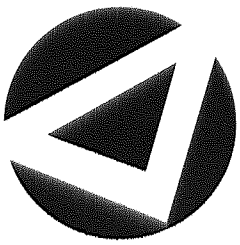
Non-linear Hierarchical Visualisation

YI SUN

Doctor of Philosophy



ASTON UNIVERSITY

September 2002

ASTON UNIVERSITY

Non-linear Hierarchical Visualisation

YI SUN

Doctor of Philosophy, 2002

Thesis Summary

This thesis applies a hierarchical latent trait model system to a large quantity of data. The motivation for it was lack of viable approaches to analyse High Throughput Screening datasets which maybe include thousands of data points with high dimensions.

High Throughput Screening (HTS) is an important tool in the pharmaceutical industry for discovering leads which can be optimised and further developed into candidate drugs. Since the development of new robotic technologies, the ability to test the activities of compounds has considerably increased in recent years. Traditional methods, looking at tables and graphical plots for analysing relationships between measured activities and the structure of compounds, have not been feasible when facing a large HTS dataset. Instead, data visualisation provides a method for analysing such large datasets, especially with high dimensions. So far, a few visualisation techniques for drug design have been developed, but most of them just cope with several properties of compounds at one time.

We believe that a latent variable model (LTM) with a non-linear mapping from the latent space to the data space is a preferred choice for visualising a complex high-dimensional data set. As a type of latent variable model, the latent trait model can deal with either continuous data or discrete data, which makes it particularly useful in this domain. In addition, with the aid of differential geometry, we can imagine the distribution of data from magnification factor and curvature plots.

Rather than obtaining the useful information just from a single plot, a hierarchical LTM arranges a set of LTMs and their corresponding plots in a tree structure. We model the whole data set with a LTM at the top level, which is broken down into clusters at deeper levels of the hierarchy. In this manner, the refined visualisation plots can be displayed in deeper levels and sub-clusters may be found. Hierarchy of LTMs is trained using expectation-maximisation (EM) algorithm to maximise its likelihood with respect to the data sample. Training proceeds interactively in a recursive fashion (top-down). The user subjectively identifies interesting regions on the visualisation plot that they would like to model in a greater detail. At each stage of hierarchical LTM construction, the EM algorithm alternates between the $E-$ and $M-$step.

Another problem that can occur when visualising a large data set is that there may be significant overlaps of data clusters. It is very difficult for the user to judge where centres of regions of interest should be put. We address this problem by employing the minimum message length technique, which can help the user to decide the optimal structure of the model.

In this thesis we also demonstrate the applicability of the hierarchy of latent trait models in the field of document data mining.

Keywords: Visualisation hierarchies, the latent trait mode, the generative topographic mapping, missing data, curvature, magnification factors, minimum message length

To my family

Acknowledgements

The first person I would like to thank is my supervisor Dr. Ian Nabney, who offered me this precious opportunity to study in the Neural Computing Research Group at Aston University. As a supervisor, he always knows exactly where my problem was when I met difficulties in our project. He is always able to give me great help and instruction. I am also grateful for helpful discussions with Dr. Peter Tiňo, who has substantial experiences on many topics in this field too. Through the valuable discussions, I continuously improve my understanding of some important conceptions in our project. Especially, I am grateful that they had discussions with me on the work appeared in Chapter 5, and I would like to thank Dr. Peter Tiňo who did the original work on the curvature of GTM manifolds.

I am indebted to Prof. David Lowe, Prof. David Saad, Dr. Manfred Opper, Dr. David Barber; I learnt a lot from taking part in their courses in the first year. I also wish to thank Dr. Dan Cornford and Dr. Laura Rebollo-Neira for organising the seminars. Their knowledge and enthusiasm led me into the wonderful world of statistical pattern recognition.

I am also grateful to my fellow students at NCRG. Lehel Csato, David Evans, Lars Hjorth, Renato Vicente and Wei Lee Woon gave me useful help on my study. They make my life in UK much easier and more interesting. Also, a sincere thank to Vicky Bond, our group assistant, for all her help.

This project is supported by Pfizer Central Research. I would like to thank Bruce Williams and Wilma Keighley for their help and support.

Finally, I would like to thank my family, for all the love, support, patience and understanding.

Contents

List of Figures

List of Tables

Declaration

This thesis describes the work carried out between October 1999 and October 2002 in the Neural Computing Research Group at Aston University under the supervision of Dr. Ian Nabney and Dr. Peter Tiňo.

This thesis has been composed by myself and has not, nor any similar dissertation, been submitted in any previous application for a degree.

Chapter 1

Introduction

This thesis is concerned with the visualisation of a large quantity of data in a high dimensional space. The approach used is a type of latent variable model, which models the probability distribution of the observed data in terms of a set of latent (hidden) variables.

The motivation behind the project was the need to analyse High Throughput Screening datasets. High Throughput Screening (HTS) is an important tool in the pharmaceutical industry for discovering novel leads: compounds which can be further developed into drug candidates. It consists of a set of automated procedures and robotic devices to perform a test for biological activity on many thousand compounds. Because of the high degree of automation, there is a corresponding demand that the analysis of the results on the large datasets be performed quickly. With such a large amount of data, visualisation is an important tool, as it provides useful information for detecting clusters, local deviations, and outliers. For HTS, the approach we developed will help to gain a better understanding of the results of multiple screens. The application of this sort of visualisation in HTS is new.

In this chapter, we introduce the motivation for the project, explain the general framework of latent variable models, and give an overview of the whole thesis.

1.1 The motivation for the project

HTS is used for detecting lead compounds, whose pharmacological properties suggest their value as a starting point for drug development. It uses automated procedures to test the activity of thousands of compounds against a molecular disease target, usually a protein. This activity is a measure of the ability of a compound to inhibit the activity of the protein. This is the first step in judging whether the compound is a potentially successful drug candidate.

The HTS process involves five steps: compound supply, assay, data capture, data analysis and sample follow-up (Zilliox, 1998). Compounds are supplied from dry samples and then these dry samples are dissolved. Microplates are used to convey the liquid samples through the whole process. On each plate there are a number of wells, such as 96 wells, 384 wells or even higher formats. Among

them, a few wells are control wells which are used to assess the reliability of the measurement processes due to the fact that a typical screen features a large number of plates measured at different times which are not subject to the same experimental conditions. Thus, a screen usually consists of several groups of a certain number of plates, with each group associated with an assay. Typically, each well on a plate contains 20 compounds, an enzyme and a substrate. An assay determines, for example, whether the enzyme is being inhibited by the test compounds. There are four types of measurements: luminometric, fluorimetric, radiometric and colorimetric measure. Thus, HTS uses signal detection instruments to measure the luminometric, fluorimetric, radiometric or colorimetric activity of the wells so that the biological activity of the mixtures of compounds can be estimated. Data analysis is the key step of the HTS process. It includes checking the control wells to ensure the quality of the data, and decision making to determine the mixture compounds whose biological activity is considered relevant (Zilliox, 1998). In the final stage, the active samples for a given target are submitted to lead optimisation, which is the complex process of refining the chemical structure of these active mixtures of compounds to improve their drug properties for use in the clinic. To optimise leads, researchers usually employ a combination of empirical, combinatorial and rational methods in a continuously multi-step process.

In the last decade, the technology of screening compounds against a wide range of therapeutic targets has improved rapidly. Now a screen can feature a large number of plates. For example, processing speeds of some HTS system can exceed 1000 plates per hour, depending on configuration[1]. With this increased efficiency, the activities of thousands of samples may be recorded. However, hitherto most of the analysis of this data has considered a single screen at a time because there are no effective analysis tools available to cope with the extremely large amounts of information. It is hard to find regularities when looking at raw data, e.g. tables of these samples. We therefore need visualisation techniques to help us.

There are a few visualisation techniques for drug design in the current stage. For example, Spotfire[2] is a *de facto* standard for data-mining visualisation in chemical applications. In addition, Roberts *et al.* (2000) have described the LeadScope application, where 2D bar-charts are provided for several variants. The user is allowed to focus on interesting molecular property ranges and common structural features. So far, in general, most of available tools depend on scatter plots, bar-charts, and so on.

This project, funded by Pfizer Central Research Institute, aims at helping scientists to understand HTS results better through the use of advanced visualisation approaches. Instead of just selecting a few properties as the research object each time, we are interested in working in the whole property space. The application of the approach we have developed is not limited to HTS data; as will be seen in the later chapters, it can also be applied in other fields, e.g. text mining. Generally, visualisation plots provide a means of understanding multivariate data.

[1] `http://www.obpw.com/high.htm`
[2] `http://www.spotfire.com/products/`

		manifest variables	
		metrical	categorical
latent variables	metrical	factor analysis	latent trait analysis
	categorical	latent profiles analysis	latent class analysis

Table 1.1: Classification of latent variable models

1.2 Latent variable models

Usually, compounds are described by many different properties which may be descriptors of chemical structure, molecular properties, or physical properties. Since the human visual ability is limited to two or three dimensions, we have to search for visualisation approaches which not only reduce the dimensionality of the data but also capture the intrinsic structure in the data with as little loss of information as possible.

Latent variable models are a common choice for solving this sort of problem. A statistical model specifies the joint distribution of a set of random variables and it becomes a latent variable model when some of these variables – the latent variables – are unobservable (Bartholomew and Knott, 1999). The observed variables are also called *manifest* variables in statistics.

We distinguish variables of *metrical* (continuous) or *categorical* (discrete) type. A latent variable model can be specified as one of 4 ways, as shown in Table 1.1.

Let $\mathbf{t}$ denote the observed variables in a D-dimensional data space $\mathcal{D}: \mathcal{R}^D$ and $\mathbf{x}$ latent variables in the L dimensional space $\mathcal{H}: \mathcal{R}^L$. We shall assume that $L < D$; usually for visualisation, $L = 2$ or 3. As only $\mathbf{t}$ can be observed, we integrate out the latent variables $\mathbf{x}$ to obtain the marginal distribution of the observed data, which is given by

$$p(\mathbf{t}) = \int p(\mathbf{x})p(\mathbf{t}|\mathbf{x})\, d\mathbf{x}, \tag{1.1}$$

where $p(\mathbf{x})$ is the prior distribution of $\mathbf{x}$ and $p(\mathbf{t}|\mathbf{x})$ is the conditional distribution of $\mathbf{t}$ given $\mathbf{x}$.

From equation (1.1), we see that a latent variable model is defined by two parts: the prior distribution of the latent variable and the conditional distribution of data given the latent variable, which sets up a probabilistic relationship between the two spaces. By defining a specific prior distribution $p(\mathbf{x})$ and conditional distribution $p(\mathbf{t}|\mathbf{x})$, which is also known as a noise model, we obtain different kinds of latent variable models.

Let us consider a given dataset $\zeta = \{\mathbf{t}_n\}_{n=1,\ldots,N}$. For the purpose of visualisation, we want to map each data point $\mathbf{t}_n$ to a corresponding point in the latent space. By using Bayes' theorem, we have

$$p(\mathbf{x}|\mathbf{t}_n) = \frac{p(\mathbf{x})p(\mathbf{t}_n|\mathbf{x})}{p(\mathbf{t}_n)}, \tag{1.2}$$

which is the posterior latent density. To visualise a whole dataset in a single plot, we need to find a statistic to summarise this distribution. Two possibilities are the posterior mode, given by

$$\text{mode} = \underset{k}{\text{armax}}\, p(\mathbf{x}_k|\mathbf{t}_n); \tag{1.3}$$

or the posterior mean, given by

$$\langle \mathbf{x}|\mathbf{t}_n \rangle = \int p(\mathbf{x}|\mathbf{t}_n)\mathbf{x}\, d\mathbf{x}. \tag{1.4}$$

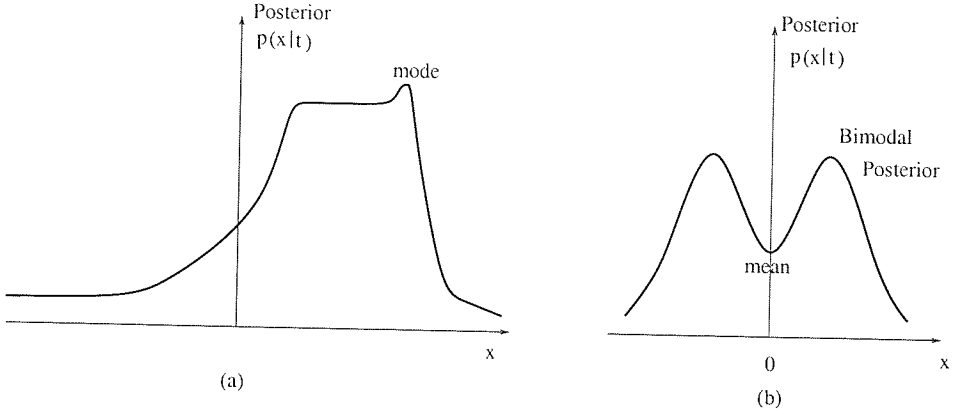


Figure 1.1: (a) A posterior density with a plateau and a peak at the end; (b) A posterior density with two peaks.

Note that both the mode and the mean can give misleading results. For example, Figure 1.1 (a) shows a posterior mode at the edge of a plateau. However, it appears that there is more probability associated with the area around the center of the plateau. On the other hand, if $p(\mathbf{x}|\mathbf{t})$ is multimodal for some point $\mathbf{t}$, as illustrated in Figure 1.1 (b), then the mean of this multimodal distribution can be a low probability point, in which case the posterior mean is inappropriate.

Several types of latent variable models have been proposed.

- Probabilistic principal component analysers (Tipping and Bishop, 1999), which put the traditional PCA into a probabilistic framework involving a linear transformation between the latent space and the data space.

- The generative topographic mapping (GTM) (Bishop *et al.*, 1998), which introduced a Gaussian noise model with a form of non-linear mapping.

- The latent trait model (LTM) (Kabán and Girolami, 2001), whose noise distribution is one from the exponential family.

We will further describe these models in the next chapter.

It is known, however, that a single two-dimensional projection is unlikely to reveal all of the interesting structures in the high-dimensional data space. This has motivated the development of algorithms which are based on a hierarchical mixture of latent variable models. For example, Bishop and Tipping (1998) developed hierarchical mixtures of probabilistic principal component analysers. As a further improvement, a hierarchical mixture of generative topographic mappings was proposed by Tiňo and Nabney (2002). The introduction of the latent trait model will offer the possibility of

developing a much more generalised hierarchical mixture system to deal with either discrete data or continuous data.

1.3 The structure of this thesis

Chapter 2 We review mixture models and several techniques of visualisation based on latent variable models and data topography (i.e. distance preserving) respectively. Experimental results suggest that both the generative topographic mapping and the generalised latent trait model are suitable tools for our task.

Chapter 3 Since we are more interested in non-linear latent variable models, we concentrate our attention on the generative topographic mapping. We have developed the GTM along two directions. One is to visualise the *curvature* of a manifold when the data is mapped into the data space from the latent space, by means of ideas from differential geometry. The other is to cope with missing data variables in the training process. In many applications the input data is incomplete. For example, in some industrial experiments some records are missing due to mechanical breakdowns. Therefore it is important to use all the available values and to reconstruct the missing values. Furthermore, we show how to utilise class information as a constraint to help recover the missing values.

Chapter 4 We address the architecture of a hierarchical visualisation system, in which the basic building block is the generalised latent trait model developed by Kabán and Girolami (2001). Since latent variable models (LVMs) are probabilistic models, it is straightforward to develop LTM in a hierarchical structure by using an extension of the *expectation-maximisation*, or EM, algorithm (Dempster *et al.*, 1977). The system provides an *interactive* mode so that the user can guide the training of the hierarchy by choosing centres of separated-well clusters, which are used to initialise the next level of the hierarchy. We also present experimental results on several different real datasets to show that our system can be employed for both discrete and continuous data.

Chapter 5 We further develop the hierarchical visualisation system by adding an *automatic* mode for initialisation. The original work is proposed by Figueiredo and Jain (2002). When it is difficult to determine *centres of interest*, an approach based on the minimum message length criterion is used to set the number of components and their corresponding position. Experimental results show that this method can work well for not only simple mixture models but deeper levels of a hierarchy.

Chapter 6 We end the whole thesis by summarising several important issues in each chapter. We address two main possible applications and suggest directions for further work as well.

1.4 Publications based on work in this thesis

This thesis involves and complements the contents of earlier publications:

- P. Tiňo, I. T. Nabney, Y. Sun: *Using Directional Curvatures to Visualise Folding Patterns of the GTM Projection Manifolds*, presented at the International Conference on Artificial Neural Networks (ICANN), (eds) G. Dorffner, H. Bischof and K. Hornik. pp. 421-428, Springer-Verlag, 2001. (Chapter 3)

- Y. Sun, P. Tiňo, I. T. Nabney: *Visualisation of Incomplete Data Using Class Information Constraints*, to appear in Uncertainty in Geometric Computations, 2002. (Chapter 3)

- P. Tiňo, I. T. Nabney, Y. Sun, B. S. Williams: *A Principled Approach to Interactive Hierarchical Non-Linear Visualisation of High-Dimensional Data*, to appear in proceedings of Interface'01-Frontiers in Data Mining and Bioinformatics, 2002. (Chapter 4)

- P. Tiňo, Y. Sun, I. T. Nabney: *Semi-Supervised Construction of General Visualisation Hierarchies*, to appear in Proceedings of the 2002 International Conference on Artificial Intelligence, 2002. (Chapter 5)

A technical report has been finished, based on the results of Chapter 4 and Chapter 5. Y. Sun, P. Tiňo, A. Kabán and I. T. Nabney: *Semi-Supervised Learning of Hierarchical Latent trait Models for Data Visualisation.* Technical Report NCRG/2002/012, NCRG, Aston University.

1.5 Datasets used in this thesis

1.5.1 HTS data

In co-operation with Pfizer Central Research we performed a series of experiments with a dataset of molecular compounds which are labelled by their biological activity against 4 different targets. If a compound shows no activity across all 4 screens, it is labeled as *0-active* class. Otherwise, if it has activity in one screen, it is labelled as *1-active* class; two screens, *2-active* class; three screens, *3-active* class. In this set there is no compound which is active in all 4 screens. The dataset contains 1399 compounds with 15 variables. It does not contain any descriptor of the chemical structure, but contains features for representing whole molecule properties for the chemical entities. They are dose value, ring index, SLogP, MlogP, number of hydrogen bonds, parent molecular weight, number of atoms, measurement of active. Both SLogP and MlogP reflect hydrophobicity of molecule, and they are measurement of the logarithm of the octanol/water partition coefficient.

1.5.2 Oil flow data

The oil flow dataset[3] is 12-dimensional and is generated by a physics-based simulation of a non-invasive monitoring system used to determine the quantity of oil in a multi-phase pipeline containing a mixture of oil, water and gas. Six pairs of γ-beams are sent through the pipe. The two beams in each pair are with different wave length. The path length through water and oil can be computed by measurements of the attenuation of these beams. It includes 1000 data points, which are classified into three classes, namely *homogeneous*, *annular* and *laminar*. These classes represent different configurations of flow in the pipe, illustrated in Figure 1.2.

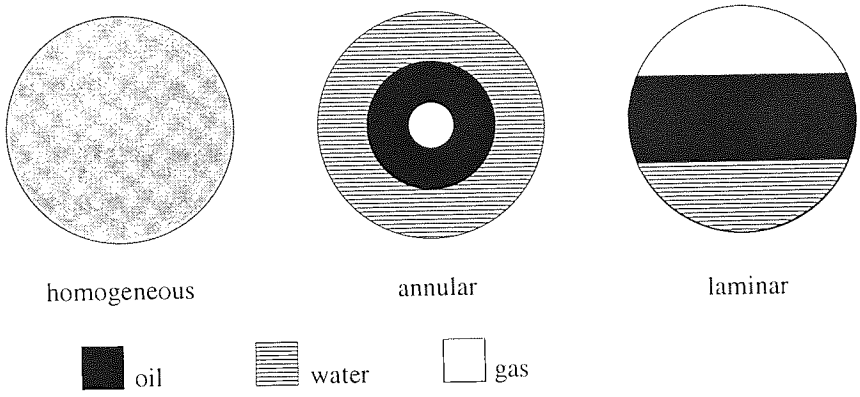


Figure 1.2: The configurations of flow in the pipe, from left to right, homogeneous, annular and laminar.

1.5.3 Image segmentation data

An image dataset[4] was obtained by randomly sampling patches of 3x3 pixels from a database of 7 outdoor images. The patches are characterised by 19 continuous attributes and classified into 7 classes: *brickface*, *sky*, *foliage*, *cement*, *window*, *path* and *grass*. It includes 2310 data points with 330 instances per class. We merged the original 7 classes into 4 classes: *cement + path*, *brickface + window*, *grass + foliage* and *sky*. Attributes are listed as follows:

1. region-centroid-col: the column of the centre pixel of the region.
2. region-centroid-row: the row of the centre pixel of the region.
3. region-pixel-count: the number of pixels in a region = 9.
4. short-line-density-5: the results of a line extraction algorithm that counts how many lines of length 5 (any orientation)with low contrast, less than or equal to 5, go through the region.
5. short-line-density-2: same as short-line-density-5 but counts lines of high contrast, greater than 5.
6. vedge-mean: measure the contrast of horizontally adjacent pixels in the region. There are 6, the mean and standard deviation are given. This attribute is used as a vertical edge detector.
7. vedge-sd: (see 6).
8. hedge-mean: measures the contrast of vertically adjacnt pixels. Used for horizontal line detection.
9. hedge-sd: (see 8).
10. intensity-mean: the average over the region of $(R + G + B)/3$.

[3]The oil flow dataset can be accessed from `http://www.ncrg.aston.ac.uk/GTM/3PhaseData.html`.

[4]The image segmentation dataset can be accessed from `http://www.ics.uci.edu/~mlearn/MLSummary.html`.

11. rawred-mean: the average over the region of the R value.
12. rawblue-mean: the average over the region of the B value.
13. rawgreen-mean: the average over the region of the G value.
14. exred-mean: measure the excess red: $2R - (G + B)$.
15. exblue-mean: measure the excess blue: $2B - (G + R)$.
16. exgreen-mean: measure the excess green: $2G - (R + B)$.
17. value-mean: 3-d non-linear transformation of RGB.
18. saturation-mean: (see 17).
19. hue-mean: (see 17).

When this dataset is employed in this thesis, we simply delete the third property since it is constant over the whole set. Note that for all work in this thesis properties 1 and 2 are included in analysis. With hindsight these location features should not have been involved since they are not intrinsic scene properties (and thus would not be used in image segmentation), but their inclusion does not affect the conclusions of this thesis.

1.5.4 Document data

A text-collection of 8000 documents is grouped into 10 topic classes from a newsgroup[5] text corpus. 800 instances were taken from each topic. The initial pre-processing, word-stemming and removal of 'stop-words' were done by using the Bow toolkit[6]. The instances are binary encoded over a dictionary of $D = 100$ words (Sahami, 1998), which was generated by mutual information with the class labels. The software also throws out *stop words* according to a pre-defined list. *Stop words* are those frequently used words which do not contribute to the meaning of the text, such as "and", "so", and so on. In addition, by using *stemming* option, all the words are reduced to their roots. For example, "learning" and "learnable" are considered to represent the same word "learn".

Note that this is a synthetic dataset and much larger vocabularies and non binary representations are employed in real document analysis and mining. A couple of references can be viewed in (van Rijsbergen, 1979; Deerwester *et al.*, 1990).

1.5.5 Yeast dataset

1484 data points for the Yeast dataset[7] with 6 attributes[8] are classified into 10 classes. The classes are cellular localisation sites of proteins, namely *CYT (cytosolic or cytoskeletal), NUC (nuclear), MIT (mitochondrial), ME3 (membrane protein, no N-terminal signal), ME2 (membrane protein, uncleaved signal), ME1 (membrane protein, cleaved signal), EXC (extracellular), VAC (vacuolar), POX (peroxisomal) and ERL (endoplasmic reticulum lumen)*, where each class includes 463, 429, 244, 163, 51, 44, 37, 30, 20, 5 data points, respectively. The attributes are from different rules for signal sequence recognition. For instance, the first two attributes are named "mcg" and "gvh" respectively. They are McGeoch's method (Mcgeoch, 1985) and von Heijne's method (Heijne, 1986) for signal sequence recognition.

[5] `http://www.cs.cmu.edu/~textlearning`
[6] `http://www-2.cs.cmu.edu/~mccalum/bow`
[7] The yeast dataset can be accessed from `http://www.ics.uci.edu/~mlearn/MLSummary.html`.
[8] The original data is 8-dimensional. Two of the dimensions are effectively constant and were deleted.

The data points were generated as follows: input sequences were obtained by receiving the information of an amino acid sequence and its source origin, formed by standard one-letter code for 20 amino acids. Then, they were analysed by applying the stored rules for various sequence features of known protein sorting signals. For example, McGeoch's method considers the N-terminal positively-charged region (N-region) and the central hydrophobic region (H-region) of signal sequences. A discriminant score is calculated from values of length of H-region, peak value of H-region, and net charge of N-region. These results are summarised in "mcg". Finally, the possibility for the input protein to be localised at each candidate site with additional information were reported [9].

1.6 Notation

We denote scalar values in lower case, while vectors and matrices are denoted by lower case bold letters and upper case bold letters, respectively. The determinant of a matrix $\mathbf{I}$ is denoted by $|\mathbf{I}|$. The symbols used for the most commonly occurring quantities in the thesis are listed in Table 1.2.

D	number of data dimensions;
d	data dimension label;
L	number of latent dimensions;
l	latent dimension label;
N	number of data points;
n	data label;
K	number of latent data points;
$\mathbf{T}$	dataset stored as a $D \times N$ matrix;
$\boldsymbol{X}$	data points in the latent space stored as an $L \times K$ matrix;
$\mathcal{I}$	observed Fisher information matrix;
$\mathbf{I}(\boldsymbol{\theta})$	expected Fisher information matrix;
$\mathbf{I}$	an identity matrix;
$\mathbf{W}^T$	transpose of matrix $\mathbf{W}$;
ζ	a collection of data points;
P	probability;
p	probability density function;
$\boldsymbol{g}(\cdot)$	link function;
$\mathcal{M}$	a sub-model in a hierarchical tree or an arbitrary model;
a	index of a mixture model;
$\boldsymbol{\theta}$	parameter vector of a mixture model;
$\mathcal{V}$	a mixture object;
$\mathcal{L}$	likelihood of $\boldsymbol{\theta}$ for the given ζ;
$\jmath$	expected message length;
$\mathcal{D}_{KL}$	*Kullback-Leibler* divergence;
$\langle\cdot\rangle$	expectation

Table 1.2: Notation in the thesis.

[9] http://psort.ims.u-tokyo.ac.jp/

Chapter 2

Latent Variable Models and Visualisation

In this chapter we review two types of visualisation technique. One arises from latent variable models, while the other is based on topography (i.e. distance preserving). We describe probabilistic principal component analysis (PPCA), the generative topographic mapping (GTM) and the latent trait model (LTM) based on the principle of latent variable models in a mixture model framework. It is therefore straightforward to develop an EM training algorithm, which provides a simple and practical method for estimating the mixture parameters. As an example of a topographic visualisation technique we introduce Neuroscale. Finally, we briefly discuess principal curves and surfaces, which are another important non-linear model used for visualisation.

2.1 Introduction

As addressed in Chapter 1 the goal of LVMs is to model the distribution $p(\mathbf{t})$ of the data $\mathbf{t}$ in a D-dimensional space in terms L latent variables $\mathbf{x}$. It is a density model, modelling an unconditional probability density given a finite set of data points $\{\mathbf{t}_n\}_{n=1,\ldots,N}$ drawn from that density function. The approach to density estimation we shall focus on is the mixture model, which belongs to the class of semi-parametric models as it defines a very general class of functional forms where the number of adaptive parameters can be increased by considering more components in the mixture to construct more flexible models.

The mixture model can be viewed, in turn, as a type of latent variable models since components are directly related to latent variables. The Gaussian mixture model (GMM) is a simple mixture model. In fact, both PPCA and the GTM are Gaussian mixture models.

In the next section, we review mixture models and a general procedure for fitting mixtures, known as the EM algorithm (Dempster *et al.*, 1977). A simple Gaussian mixture model is introduced. An important and easy-to-use visualisation method, PCA, is discussed in section 2.3. Section 2.4 is about

probabilistic PCA, which involves a linear mapping between the latent space and the data space. As a more complex latent variable model, we consider the generative topographic mapping (GTM), which includes a non-linear transformation, in Section 2.5. However, the GTM is only able to cope with continuous observed data. In order to deal with discrete data in data space, a latent trait model which uses a member of the exponential family as a noise model appears in section 2.6. In the last section, Neuroscale, which is based on data topography, is discussed.

2.2 Mixture models

Before we further discuss some specific LVMs, we shall restrict our attention to mixture models, which are the main density modelling technique. As will be seen, LVMs introduced in later sections have a close relationship with mixture models.

2.2.1 Mixture models

Consider the problem of modelling a probability density function $p(\mathbf{t})$ given a finite number of data points $\{\mathbf{t}_n\}_{n=1,\ldots,N}$. The density of the data can be approximated using a model which is a linear combination of simple *component* densities $p(\mathbf{t}|\boldsymbol{\theta}_a)$ (Bishop, 1995):

$$p(\mathbf{t}) = \sum_{a=1}^{A} P(a)p(\mathbf{t}|\boldsymbol{\theta}_a), \tag{2.1}$$

where $P(a)$ are the *mixing coefficients*, and satisfy the properties

$$\sum_{a=1}^{A} P(a) = 1, \qquad 0 \leq P(a) \leq 1, \tag{2.2}$$

which guarantee that $p(\mathbf{t})$ is a valid density function. The parameters of the mixture model are $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_A)$ and P. In this thesis, we assume that all components have the same functional form, and each is specified by the parameter vector $\boldsymbol{\theta}_a$. For detailed and comprehensive accounts of mixture models, see (McLachlan and Basford, 1988; McLachlan and Peel, 2000; Titterington *et al.*, 1985).

A mixture model is able to represent arbitrarily complex probability density functions (pdf's), provided that the model has enough components and the parameters of the model are selected appropriately. It is also a choice for representing complex class-conditional pdf's which are likelihood functions in Bayesian supervised learning frameworks (Hastie and Tibshirani, 1996; Hinton *et al.*, 1997; Streit and Luginbuhl, 1994), and priors for Bayesian parameter estimation (Dalal and Hall, 1983) as well. Mixture models can also be used to cluster data in an unsupervised way (Jain and Dubes, 1988; Jain *et al.*, 2000; McLachlan and Basford, 1988; McLachlan and Peel, 2000; Titterington *et al.*, 1985).

2.2.2 EM algorithm for mixture models

The next step is to estimate the parameters of a mixture model from a set of data. Consider a dataset of N vectors $\zeta = \{\mathbf{t}_n\}_{n=1,\ldots,N}$. If we assume that these vectors are drawn independently from the

distribution $p(\mathbf{t})$, then the joint probability density of ζ is given by

$$p(\zeta|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\mathbf{t}_n) = \prod_{n=1}^{N} \left\{ \sum_{a=1}^{A} P(a) p(\mathbf{t}_n|\boldsymbol{\theta}_a) \right\} \equiv \mathcal{L}(\boldsymbol{\theta}), \tag{2.3}$$

where $\mathcal{L}(\boldsymbol{\theta})$ is referred to as the *likelihood* of $\boldsymbol{\theta}$ for given ζ.

For determining the parameters of a mixture model from a set of data, we minimise the negative log-likelihood for the dataset, which is given by (Bishop, 1995)

$$\begin{aligned} E &= -\log \mathcal{L}(\boldsymbol{\theta}) \\ &= -\sum_{n=1}^{N} \log p(\mathbf{t}_n) \\ &= -\sum_{n=1}^{N} \log \left\{ \sum_{a=1}^{A} P(a) p(\mathbf{t}_n|\boldsymbol{\theta}_a) \right\}. \end{aligned} \tag{2.4}$$

This can be regarded as an error function. A simple and practical method used to fit mixture models to observed data is the *expectation-maximisation*, or EM, algorithm (Dempster *et al.*, 1977).

In the EM algorithm, the observed data matrix $\{\mathbf{t}_n\}_{n=1,\ldots,N}$ is regarded as *incomplete data*, since a set of N discrete assignment variable vectors, $\mathcal{Z} = \{\mathbf{z}_n\}_{n=1,\ldots,N}$, are missing. An assignment variable $z_{an} = (\mathbf{z}_n)_a$ is introduced for each data point $\mathbf{t}_n$ to specify whether $\mathbf{t}_n$ was generated by the ath component. $z_{an} = 1$ if and only if the data point $\mathbf{t}_n$ was generated by the ath component of the mixture, otherwise $z_{an} = 0$.

Denote by ζ_{comp} a complete dataset including $\mathbf{t}_n$ and $\mathbf{z}_n$. The likelihood function $\mathcal{L}_{comp}(\boldsymbol{\theta})$ for the complete dataset can be written as

$$\mathcal{L}_{comp}(\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\mathbf{t}_n, \mathbf{z}_n) = \prod_{n=1}^{N} \left\{ P(\mathbf{z}_n) p(\mathbf{t}_n|\mathbf{z}_n) \right\}. \tag{2.5}$$

The negative log-likelihood for this complete dataset, having multinomial form (Mclachlan and Krishnan, 1997), is given by

$$\begin{aligned} E_{comp} &= -\log \mathcal{L}_{comp}(\boldsymbol{\theta}) \\ &= -\sum_{n=1}^{N} \log \left\{ \prod_{a=1}^{A} (P(a))^{z_{an}} \prod_{a=1}^{A} \left(p(\mathbf{t}_n|\boldsymbol{\theta}_a)\right)^{z_{an}} \right\} \\ &= -\sum_{n=1}^{N} \sum_{a=1}^{A} z_{an} \log \left\{ P(a) p(\mathbf{t}_n|\boldsymbol{\theta}_a) \right\}. \end{aligned} \tag{2.6}$$

The EM algorithm proceeds iteratively in two steps, $E-$ and $M-$steps, by treating z_{an} as missing data. The iteration starts from some initial parameters $\boldsymbol{\theta}^{(0)}$. The jth iteration is as follows:

- $E-$step

  Compute the expectation of E_{comp} with respect to the probability distribution of assignment variables $\mathbf{z}_n$. It is given by

$$\langle E_{comp} \rangle = -\sum_{n=1}^{N} \sum_{a=1}^{A} P^{(j-1)}(a|\mathbf{t}_n) \left[\log P(a) + \log p(\mathbf{t}_n|\boldsymbol{\theta}_a) \right], \tag{2.7}$$

where

$$P^{(j-1)}(a|\mathbf{t}_n) = \frac{P^{(j-1)}(a)p^{(j-1)}(\mathbf{t}_n|\boldsymbol{\theta}_a)}{\sum_{a=1}^{A} P^{(j-1)}(a)p^{(j-1)}(\mathbf{t}_n|\boldsymbol{\theta}_a)}. \tag{2.8}$$

- $M-$step

  Update parameters by minimising the expected error $\langle E_{comp} \rangle$ with respect to the parameters

$$\boldsymbol{\theta}^j = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \langle E_{comp} \rangle. \tag{2.9}$$

A variational view of the EM algorithm

Jordan *et al.* (1998) introduced a variational approach to the EM algorithm. The basic idea of variational methodology is to convert a complex problem into a simpler problem, which is generally characterised by decoupling of the degrees of freedom in the original problem. The decoupling is achieved by including additional parameters, known as variational parameters, that must be fit to the problem at hand (Jordan *et al.*, 1998).

Considering a situation in which we are unable to compute the conditional distribution $p(\mathcal{Z}|\zeta)$. In such cases, variational methodology suggests that we consider a family of approximating conditional distributions $\hat{p}(\mathcal{Z}|\zeta, \lambda)$, where λ are variational parameters.

Now we need a measure of approximation accuracy. Jordan *et al.* (1998) showed that using the Kullback-Leibler divergence (Kullback and Leibler, 1951) $\mathcal{D}_{KL}$ as a measure of approximation accuracy yields the best *lower bound* on the log-likelihood in the family of approximations $\hat{p}(\mathcal{Z}|\zeta, \lambda)$. The $\mathcal{D}_{KL}$ is defined as follows (Cover and Thomas, 1991):

$$\mathcal{D}_{KL}\left\{\hat{p}(\mathcal{Z}|\zeta, \lambda)||p(\mathcal{Z}|\zeta)\right\} = \sum_{\mathcal{Z}} \hat{p}(\mathcal{Z}|\zeta, \lambda) \log \frac{\hat{p}(\mathcal{Z}|\zeta, \lambda)}{p(\mathcal{Z}|\zeta)}. \tag{2.10}$$

They bounded the log-likelihood using Jensen's inequality as follows:

$$\begin{aligned} \log p(\zeta) &= \log \sum_{\mathcal{Z}} p(\mathcal{Z}, \zeta) \\ &= \log \sum_{\mathcal{Z}} \hat{p}(\mathcal{Z}|\zeta) \frac{p(\mathcal{Z}, \zeta)}{\hat{p}(\mathcal{Z}|\zeta)} \\ &\geq \sum_{\mathcal{Z}} \hat{p}(\mathcal{Z}|\zeta) \log \frac{p(\mathcal{Z}, \zeta)}{\hat{p}(\mathcal{Z}|\zeta)}. \end{aligned} \tag{2.11}$$

The difference between the left and right hand sides of equation (2.11) can be viewed to be the $\mathcal{D}_{KL}$ between $p(\mathcal{Z}, \zeta)$ and $\hat{p}(\mathcal{Z}|\zeta)$. Thus, a particular distribution by minimising $\mathcal{D}_{KL}$ with respect to the variational parameters is selected from the family of approximating distribution $\hat{p}$. The selected particular distribution is treated as the best approximation of $p(\mathcal{Z}|\zeta)$ in the family $\hat{p}(\mathcal{Z}|\zeta, \lambda)$.

$$\lambda^* = \underset{\lambda}{argmin} \left[\mathcal{D}_{KL}\left\{\hat{p}(\mathcal{Z}|\zeta, \lambda)||p(\mathcal{Z}|\zeta)\right\}\right]. \tag{2.12}$$

By selecting λ according to equation (2.12), we obtain the tightest lower bound.

Now let us include parameters $\boldsymbol{\theta}$ in the marginal probability $p(\zeta|\boldsymbol{\theta})$, known as the likelihood. The function

$$\mathcal{F}(\hat{p}, \boldsymbol{\theta}) = \hat{p}(\mathcal{Z}|\zeta) \log \hat{p}(\mathcal{Z}|\zeta) - \sum_{\mathcal{Z}} \hat{p}(\mathcal{Z}|\zeta) \log p(\mathcal{Z}, \zeta|\boldsymbol{\theta}) \tag{2.13}$$

is a upper bound on the negative log-likelihood ($-\log p(\zeta|\boldsymbol{\theta})$) for any probability distribution $\hat{p}(\mathcal{Z}|\zeta)$.

This suggests the following algorithm. Again the iteration starts from some initial parameters $\boldsymbol{\theta}^{(0)}$. First the bound $\mathcal{F}(\hat{p}, \boldsymbol{\theta})$ is minimised with respect to probability distribution $\hat{p}$. Second, $\hat{p}$ is fixed and minimised $\mathcal{F}(\hat{p}, \boldsymbol{\theta})$ with respect to the parameters $\boldsymbol{\theta}$. That is in the jth iteration:

- $E-$step

$$\hat{p}^j = \underset{\hat{p}}{\operatorname{argmin}} \{\mathcal{F}(\hat{p}, \boldsymbol{\theta}^{(j-1)})\}; \tag{2.14}$$

- $M-$step

$$\boldsymbol{\theta}^j = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \{\mathcal{F}(\hat{p}^j, \boldsymbol{\theta})\}. \tag{2.15}$$

This can be related to the EM algorithm by noting that the right hand side in equation (2.13) is a function of $\boldsymbol{\theta}$ only through the term $\log p(\mathcal{Z}, \zeta|\boldsymbol{\theta})$ when fixing $\hat{p}$. Thus minimising $\mathcal{F}(\hat{p}, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ in the $M-$step is equivalent to minimising the complete negative log-likelihood in the EM algorithm.

An proximal point view of the EM algorithm

A generalized proximal point algorithm (Chrétien and Hero, 2000) is defined by the following iteration

$$\boldsymbol{\theta}^j = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{ \mathcal{E}(\boldsymbol{\theta}) + \gamma_k d(\boldsymbol{\theta}, \boldsymbol{\theta}^{(j-1)}) \right\}, \tag{2.16}$$

where $\mathcal{E}(\boldsymbol{\theta})$ is some function whose minimum with respect to $\boldsymbol{\theta}$ is sought, γ_k is a sequence of positive numbers, and $d(\boldsymbol{\theta}, \boldsymbol{\theta}^j)$ is a distance-like penalty function. In this framework, the EM algorithm for fitting a mixture model can be viewed as a proximal point algorithm with $\mathcal{E}(\boldsymbol{\theta}) = -\log \mathcal{L}(\boldsymbol{\theta})$, $\gamma_k = 1$ and with the Kullback-Leibler divergence $\mathcal{D}_{KL}$ between $p(\mathbf{z}|\mathbf{t}, \boldsymbol{\theta}^j)$ given the estimated model and $p(\mathbf{z}|\mathbf{t}, \boldsymbol{\theta})$ given the true model

$$\begin{aligned} d(\boldsymbol{\theta}, \boldsymbol{\theta}^j) &= \mathcal{D}_{KL} \left\{ p(\mathbf{z}|\mathbf{t}, \boldsymbol{\theta}^j) || p(\mathbf{z}|\mathbf{t}, \boldsymbol{\theta}) \right\} \\ &= \int p(\mathbf{z}|\mathbf{t}, \boldsymbol{\theta}^j) \log \frac{p(\mathbf{z}|\mathbf{t}, \boldsymbol{\theta}^j)}{p(\mathbf{z}|\mathbf{t}, \boldsymbol{\theta})} \, d\mathbf{z}. \end{aligned} \tag{2.17}$$

Based on this framework, Celeux *et al.* (2001) proposed a component-wise EM algorithm for mixtures and proved its convergence by using the proximal point algorithm. The idea of component-wise EM algorithm is to update only one component at a time, leaving the other parameters unchanged. Assuming it is in the jth iteration, then we have

- E-step:

  Compute the posterior distribution of the ath component given the observed data by using equation (2.8).

- M-step:

  Update parameters $\boldsymbol{\theta}_a$ of the ath component;

  for $a' \neq a$, we have $\boldsymbol{\theta}_{a'}^j = \boldsymbol{\theta}_{a'}^{(j-1)}$.

Main drawbacks of EM algorithm

- Since EM is a local approach, it is highly depend on initialisation.

  The error function has a large number of local minima, many of which correspond to poor models of the true density distribution. Several methods have been proposed to solve this problem. For example, initialisation by using clustering algorithms, such as the K-means algorithm (Nabney, 2001), or using many different initial parameter sets and then selecting the model with the lowest error value.

- EM may converge to the boundary of the parameter space.

  For example, when training a Gaussian mixture model, if one mixing coefficient approaches zero, then the corresponding covariance matrix may approach singularity. This can also happen if a component centre gets very close to a data point. In practice, to avoid this problem, the covariance matrix is checked at each iteration, and dangerously small values are replaced by larger ones (Nabney, 2001).

In addition, in some situations EM algorithm shows slow convergence problem. A more detailed analyses and results can be seen in (Redner and Walker, 1984).

2.2.3 Gaussian mixture models

A Gaussian mixture model (GMM) is a frequently used tool for density estimation. It is defined as a mixture model (see equation (2.1)) with A Gaussian components. The distribution of each component a is $p(\mathbf{t}|\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a)$:

$$p(\mathbf{t}|\boldsymbol{\theta}_a) = p(\mathbf{t}|\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a) = \frac{1}{|\boldsymbol{\Sigma}_a|^{1/2}(2\pi)^{D/2}} \exp\left\{-\frac{1}{2}(\mathbf{t}-\boldsymbol{\mu}_a)^T\boldsymbol{\Sigma}_a^{-1}(\mathbf{t}-\boldsymbol{\mu}_a)\right\}, \tag{2.18}$$

where $\boldsymbol{\Sigma}_a$ is a $D \times D$ symmetric and positive-definite covariance matrix and $\boldsymbol{\mu}_a$ is a mean vector of component a.

The parameters of a GMM can be determined by using maximum likelihood estimation with the EM algorithm.

- $E-$step

  We compute the posterior component probabilities $P(a|\mathbf{t}_n)$ given by equation (2.8).

- $M-$step

  By minimising the expected error $\langle E_{comp}\rangle$ with respect to the parameters, we obtain equations for updating parameters of each Gaussian component:

$$P^j(a) = \frac{1}{N}\sum_{n=1}^{N} P^{(j-1)}(a|\mathbf{t}_n), \tag{2.19}$$

$$\boldsymbol{\mu}_a^j = \frac{\sum_{n=1}^{N} P^{(j-1)}(a|\mathbf{t}_n)\mathbf{t}_n}{\sum_{n=1}^{N} P^{(j-1)}(a|\mathbf{t}_n)}. \tag{2.20}$$

The update equation of the $M-$step for the covariance parameters depends on the type of covariance structure.

Spherical covariance matrices. The update equation is

$$(\sigma_a^j)^2 = \frac{1}{D} \frac{\sum_{n=1}^{N} P^{(j-1)}(a|\mathbf{t}_n)||\mathbf{t}_n - \boldsymbol{\mu}_a^j||^2}{\sum_{n=1}^{N} P^{(j-1)}(a|\mathbf{t}_n)}. \tag{2.21}$$

Diagonal covariance matrices. The update equation is

$$(\sigma_{d,a}^j)^2 = \frac{\sum_{n=1}^{N} P^{(j-1)}(a|\mathbf{t}_n)(t_{dn} - \mu_{d,a}^j)^2}{\sum_{n=1}^{N} P^{(j-1)}(a|\mathbf{t}_n)}, \tag{2.22}$$

where d denotes the dimension label.

Full covariance matrices. The update equation is

$$\boldsymbol{\Sigma}_a^j = \frac{\sum_{n=1}^{N} P^{(j-1)}(a|\mathbf{t}_n)(\mathbf{t}_n - \boldsymbol{\mu}_a^j)(\mathbf{t}_n - \boldsymbol{\mu}_a^j)^T}{\sum_{n=1}^{N} P^{(j-1)}(a|\mathbf{t}_n)}. \tag{2.23}$$

Consider, for example, a set of two-dimensional points generated from a density, which is shown as a surface plot in Figure 2.1. In this case, a simple mixture of two Gaussian components were supposed enough to fit the data density function. Figure 2.2 presents the result after the EM training of a two-component model.

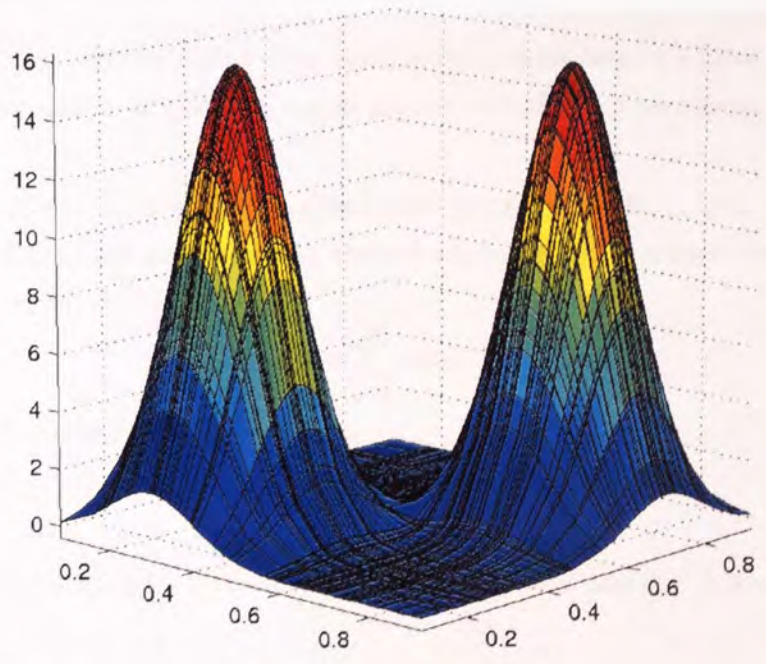


Figure 2.1: Surface plot of a probability density function in two dimensions.

GMM is a simple and useful method for modelling density function. In the latter sections, we shall see how PPCA and GTM, both are GMM, were developed to aid us to visualise data.

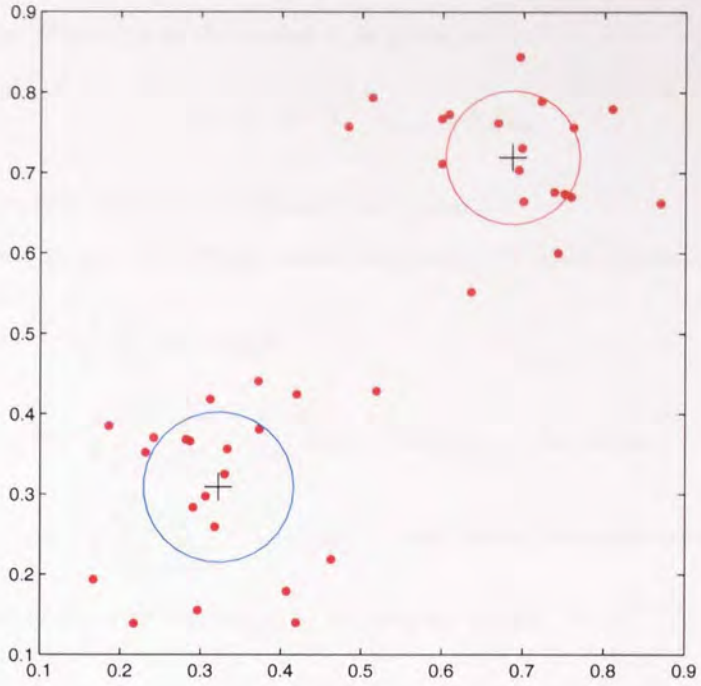


Figure 2.2: Spherical covariance mixture model. Sampled data (dots), centres (crosses) and one standard deviation error bars (lines).

2.3 Principal component analysis

Classical PCA (Bishop, 1995) is a projection method which maps data to a lower dimensional space with a linear transformation. It is one of the most popular techniques for pre-processing and visualising data.

Consider data vectors $\mathbf{t}_n$ in a D-dimensional data space. We want to map these vectors onto vectors $\mathbf{x}_n$ in a L dimensional lower space. A vector $\mathbf{t}$ can be written as a linear combination of a set of D orthonormal vectors $\mathbf{u}_d$

$$\mathbf{t} = \sum_{d=1}^{D} x_d \mathbf{u}_d, \tag{2.24}$$

where the $\mathbf{u}_d$ satisfy the orthonormality relation

$$\mathbf{u}_d^T \mathbf{u}_{d'} = \delta_{dd'}, \tag{2.25}$$

where $\delta_{dd'}$ is the Kronecker delta symbol, that is $\delta_{dd'} = 1$ if $d = d'$ and $\delta_{dd'} = 0$ otherwise. Then we have

$$x_d = \mathbf{u}_d^T \mathbf{t}. \tag{2.26}$$

Since we want a reduction in the dimensionality, we re-write equation (2.24) as

$$\tilde{\mathbf{t}} = \sum_{d=1}^{L} x_d \mathbf{u}_d + \sum_{d=L+1}^{D} h_d \mathbf{u}_d, \tag{2.27}$$

where the h_d are constants.

Our aim is to choose suitable coefficients h_d and basis vectors $\mathbf{u}_d$, so that $\tilde{\mathbf{t}}$ can approximate $\mathbf{t}$ as accurately as possible. The error in the vector $\mathbf{t}_n$ is given by

$$\mathbf{t}_n - \tilde{\mathbf{t}}_n = \sum_{d=L+1}^{D} (x_{d,n} - h_d)\mathbf{u}_d, \tag{2.28}$$

where $x_{d,n}$ denotes the dth dimension of the nth data point.

So we minimise the sum of the squares of the errors over the whole dataset, which has the form

$$\begin{aligned} E_L &= \frac{1}{2}\sum_{n=1}^{N} ||\mathbf{t}_n - \tilde{\mathbf{t}}_n||^2 \\ &= \frac{1}{2}\sum_{n=1}^{N}\sum_{d=L+1}^{D}\sum_{d'=L+1}^{D} (x_{d,n} - h_d)(x_{d',n} - h_{d'})\mathbf{u}_d^T\mathbf{u}_{d'} \\ &= \frac{1}{2}\sum_{n=1}^{N}\sum_{d=L+1}^{D} (x_{d,n} - h_d)^2 \quad \text{since the } \mathbf{u}_d \text{ are orthonormal.} \end{aligned} \tag{2.29}$$

Setting the derivative of E_L with respect to h_d to zero, we obtain

$$h_d = \frac{1}{N}\sum_{n=1}^{N} x_{d,n} = \mathbf{u}_d^T\hat{\mathbf{t}}, \tag{2.30}$$

where

$$\hat{\mathbf{t}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{t}_n. \tag{2.31}$$

Using equations (2.26) and (2.30), the error function (2.29) can be rewritten as

$$E_L = \frac{1}{2}\sum_{d=L+1}^{D}\sum_{n=1}^{N}\{\mathbf{u}_d^T(\mathbf{t} - \hat{\mathbf{t}})\}^2 = \frac{1}{2}\sum_{d=L+1}^{D} \mathbf{u}_d^T\mathbf{\Sigma}\mathbf{u}_d, \tag{2.32}$$

where $\mathbf{\Sigma}$ is the covariance matrix of $\{\mathbf{t}_n\}_{n=1,\ldots,N}$ and is given by

$$\mathbf{\Sigma} = \sum_{n=1}^{N}(\mathbf{t}_n - \hat{\mathbf{t}})(\mathbf{t}_n - \hat{\mathbf{t}})^T. \tag{2.33}$$

Minimising equation (2.32) with respect to $\mathbf{u}_d$, it can be shown that when the basis vectors $\mathbf{u}_d$ are the eigenvectors of $\mathbf{\Sigma}$, which is $\mathbf{\Sigma}\mathbf{u}_d = \lambda_d\mathbf{u}_d$, we have the minimum error in the form $E_M = \frac{1}{2}\sum_{d=L+1}^{D}\lambda_d$. This suggests that the minimum error is obtained by selecting the $D - L$ smallest eigenvalues.

In practice, in order to project $\mathbf{t}_n$ onto a lower L dimensional space, the mean of the input vectors $\mathbf{t}_n$ first is calculated and then subtracted. Then the covariance matrix is computed and its eigenvectors and eigenvalues are obtained. L eigenvectors which correspond to the L largest eigenvalues are kept. Then the vectors $\mathbf{t}_n$ are mapped onto these eigenvectors to generate components of $\mathbf{x}_n$ in the L dimensional space.

Although PCA is easy and fast to calculate, it suffers from two drawbacks:

- It can only find a linear subspace so it cannot deal properly with data lying on non-linear manifolds.

- One doesn't know how many principal components to keep, although some rules of thumb are applied in practice, where most of them need probabilistic model to apply Bayesian and related structure selection criteria. We will introduce more details in the next section.

The experimental results

Figure 2.3 shows a visualisation plot of the oil dataset (see section 1.5.2). We project it onto the first two principal components after subtracting the data mean. Note the *Laminar* class is separated into five distinct clusters, while the other two classes are more tightly grouped. It suggests that this linear map gives a reasonable class separation.

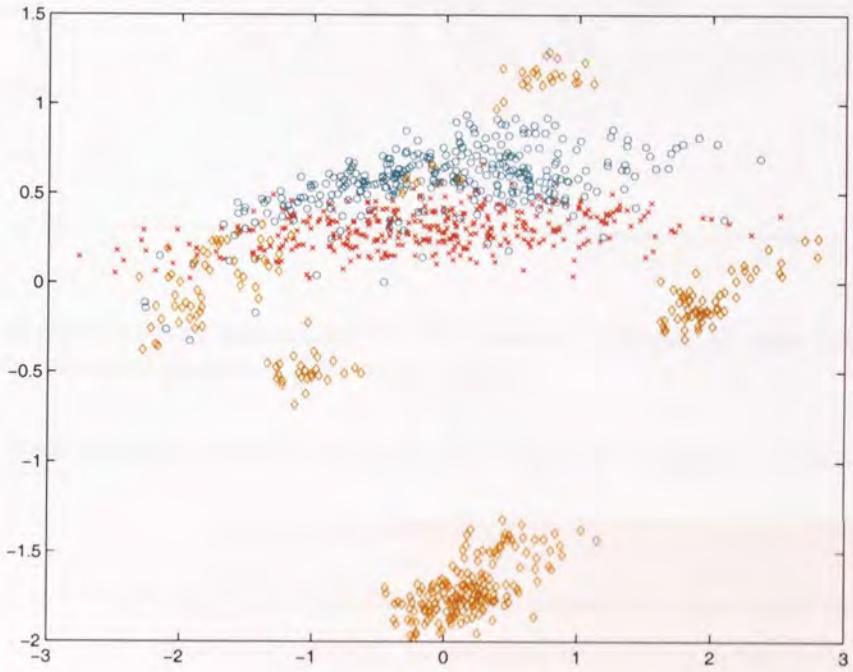


Figure 2.3: Projection of oil data using PCA. Three classes are shown: Homogeneous (crosses), annular (circles) and laminar (diamond signs).

A projection of the HTS dataset (see section 1.5.1) using PCA is presented in the Figure 2.4. As viewed, there are significant overlaps of different classes. Since PCA employs a simple linear transformation to map the data into a space of lower dimensionality, it is difficult to clearly reflect complex relationships between points. It suggests that there are more complex non-linear structures in this dataset, so using PCA to study the HTS data is not a suitable choice.

2.4 Probabilistic principal component analysis

2.4.1 Probabilistic PCA

Standard PCA does not provide a generative model (i.e. a density model) for data. Thus it is limited in comparison with density estimation techniques. Tipping and Bishop (1999) developed probabilistic PCA (PPCA) by using a latent variable model formulated within a maximum likelihood framework.

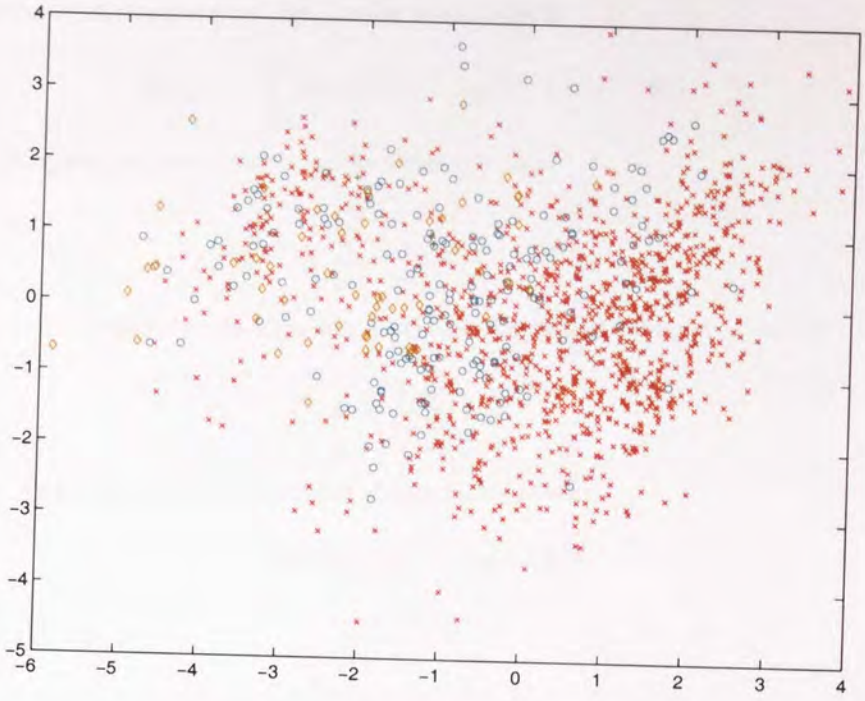


Figure 2.4: Projection of screen data using PCA. Four classes are shown: *0-active* (crosses), *1-active* (circles), *2-active* (diamond signs) and *3-active* (plus signs).

Consider a linear mapping from points $\mathbf{x}$ in the latent space $\mathcal{H}$ to points $\mathbf{t}$ in the data space $\mathcal{D}$,

$$\mathbf{t} = \mathbf{W}\mathbf{x} + \boldsymbol{\mu} + \epsilon, \tag{2.34}$$

where $\mathbf{W}$ is a $D \times L$ weight matrix, $\boldsymbol{\mu}$ is a constant which permits the data model to have non-zero mean, and ϵ is an $\mathbf{x}$-independent noise process. As discussed in section 1.2, two parts are needed to set up a latent variable model.

- The conditional distribution of the data given the latent variables. In this case, an isotropic noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ is defined. This means that

$$p(\mathbf{t}|\mathbf{x}) = (2\pi\sigma^2)^{-D/2} \exp\left\{-\frac{1}{2\sigma^2}||\mathbf{t} - \mathbf{W}\mathbf{x} - \boldsymbol{\mu}||^2\right\}. \tag{2.35}$$

- The prior distribution over the latent variables. In this framework a Gaussian prior with unit variance is given by

$$p(\mathbf{x}) = (2\pi)^{-L/2} \exp\left\{-\frac{1}{2}\mathbf{x}^T\mathbf{x}\right\}. \tag{2.36}$$

The marginal distribution of the observed data $\mathbf{t}$, obtained by integrating out the latent variable $\mathbf{x}$, is also Gaussian:

$$\begin{aligned} p(\mathbf{t}) &= \int p(\mathbf{t}|\mathbf{x})p(\mathbf{x})\, d\mathbf{x} \\ &= (2\pi)^{-D/2}|\mathbf{C}|^{-1/2} \exp\left\{-\frac{1}{2}(\mathbf{t} - \boldsymbol{\mu})^T\mathbf{C}^{-1}(\mathbf{t} - \boldsymbol{\mu})\right\}, \end{aligned} \tag{2.37}$$

where $\mathbf{C} = \sigma^2\mathbf{I} + \mathbf{W}\mathbf{W}^T$ is the model covariance.

The log-likelihood of the observed data under this model is

$$\log \mathcal{L} = -\frac{N}{2}\{D\log(2P) + \log|\mathbf{C}| + \mathrm{tr}(\mathbf{C}^{-1}\mathbf{S})\}, \tag{2.38}$$

where $\mathbf{S}$ is the sample covariance matrix of the observed data,

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{t}_n - \boldsymbol{\mu})(\mathbf{t}_n - \boldsymbol{\mu})^T, \tag{2.39}$$

provided that $\boldsymbol{\mu}$ is the sample mean from the maximum likelihood estimate, given by

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{t}_n. \tag{2.40}$$

With maximum likelihood estimates, we also obtain parameters

$$\mathbf{W} = \boldsymbol{U}_L(\Lambda_L - \sigma^2\mathbf{I})^{1/2}\boldsymbol{J}, \tag{2.41}$$

and

$$\sigma^2 = \frac{1}{D-L}\sum_{l=L+1}^{D}\lambda_l, \tag{2.42}$$

where the L column vectors in the $D \times L$ matrix $\mathbf{U}_L$ are the principal L eigenvectors of $\mathbf{S}$, Λ_L is an $L \times L$ diagonal matrix with corresponding eigenvalues, $\boldsymbol{J}$ is an arbitrary $L \times L$ orthogonal rotation matrix, and $\lambda_{l+1}, \ldots, \lambda_D$ are the smallest eigenvalues of $\mathbf{S}$. σ^2 can be regarded as the average variance "lost" per discarded dimension. Because the rotation matrix $\boldsymbol{J}$ is arbitrary, if $\mathbf{W}$ is computed by an eigendecomposition of $\mathbf{S}$ (which is much more computationally efficient than using the EM algorithm), then $\boldsymbol{J}$ may be assumed to be the identity matrix (Nabney, 2001).

Usually, the representation $\tilde{\mathbf{x}}_n$ can be obtained by using the standard projection, which is given by $\tilde{\mathbf{x}}_n = \mathbf{W}^T(\mathbf{t}_n - \boldsymbol{\mu})$, without loss of information. As an alternative to the standard PCA projection, the *posterior mean* can be used (see section 1.2). By using Bayes' theorem, the *posterior* distribution of the latent variables $\mathbf{x}$ has the form

$$p(\mathbf{x}|\mathbf{t}) \sim \mathcal{N}\left(\mathbf{M}^{-1}\mathbf{W}^T(\mathbf{t} - \boldsymbol{\mu}), \sigma^2\mathbf{M}^{-1}\right), \tag{2.43}$$

where

$$\mathbf{M}^{-1} = (\sigma^2\mathbf{I} + \mathbf{W}^T\mathbf{W})^{-1}. \tag{2.44}$$

Minka's automatic choice of dimensionality for PCA

A central issue in PCA is selecting the number of principal components. Minka (2000) showed how to use Bayesian model selection to determine the dimensionality of the latent space.

To choose the subspace dimensionality L, the probability of the data from a set $\zeta = \{\mathbf{t}_1, \ldots, \mathbf{t}_N\}$ for each possible dimensionality is computed. The probability of the data given the model $\mathcal{M}$ is calculated by integrating over the unknown parameter values $\boldsymbol{\theta}$ in that model:

$$p(\zeta|\mathcal{M}) = \int_{\boldsymbol{\theta}} p(\zeta|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{M})\, d\boldsymbol{\theta}. \tag{2.45}$$

This quantity is called the *evidence* for model $\mathcal{M}$. More details on discussing Bayesian model selection can be seen in (MacKay, 1995).

By using conjugate priors for the eigenvectors, eigenvalues of the sample covariance matrix $\mathbf{S}$ and noise level, choosing an appropriate parameterization and applying Laplace's method, Minka (2000) shows that the evidence for a PCA model with L principal components is given by

$$p(\zeta|L) \approx p(\mathbf{U}_L) \left(\prod_{l=1}^{L} \lambda_l \right)^{-N/2} (\sigma^2)^{-N(D-L)/2} (2P)^{(m+L)/2} |\mathbf{H}_Z|^{-1/2} N^{-L/2}, \tag{2.46}$$

where $m = DL - L(L+1)/2$ and

$$p(\mathbf{U}_L) = 2^{-L} \prod_{l=1}^{L} \Gamma((D-l+1)/2) P^{-(D-l+1)/2}, \tag{2.47}$$

where $\Gamma()$ is the Gamma function and

$$|\mathbf{H}_Z| = \prod_{l=1}^{L} \prod_{d=l+1}^{D} (\hat{\lambda}_d^{-1} - \hat{\lambda}_l^{-1})(\lambda_l - \lambda_d) N, \tag{2.48}$$

and λ_d are the eigenvalues from PCA, and $\hat{\lambda}_d$ are identical except for $d > L$ where $\hat{\lambda}_d = (1/(D-K)) \sum_{d=L+1}^{D} \lambda_d$.

Minka's experiments show this is an accurate and consistent model order criterion.

2.4.2 Mixture of PPCA

Because PCA defines a global linear transformation of data, it is a rather limited method to model a complex non-linear structure, as is shown in Figure 2.4. However, since PCA was extended to probabilistic PCA which is a Gaussian distribution of a particular form, it is reasonable to combine multiple PCAs in a mixture of such models (Tipping and Bishop, 1999). In this way, the mixture of PPCAs may model a non-linear structure by using a set of local linear models. The parameters can be determined using the EM algorithm.

Again the corresponding density model takes the form of equation (2.1). Each component is an independent latent variable model, PPCA, with parameters $\boldsymbol{\mu}_a, \mathbf{W}_a$ and σ_a^2. $P(\mathbf{t}|\boldsymbol{\theta}_a)$ is given by equation (2.37).

- $E-$step

  We compute the *responsibility* R_{an} of component a for generating data point $\mathbf{t}_n$

  $$R_{an} = P(a|\mathbf{t}_n) = \frac{P(a)p(\mathbf{t}_n|\boldsymbol{\theta}_a)}{p(\mathbf{t}_n)}. \tag{2.49}$$

- $M-$step

  The computation of $P^j(a)$ and $\boldsymbol{\mu}_a^j$ is again the same as in the Gaussian mixture model. The covariance matrix is re-computed for data weighted by the responsibility of the corresponding component.

  $$\mathbf{S}_a = \frac{1}{P^j(a)N} \sum_{n=1}^{N} R_{an}(\mathbf{t}_n - \boldsymbol{\mu}_a^j)(\mathbf{t}_n - \boldsymbol{\mu}_a^j)^T. \tag{2.50}$$

Then it is intuitive that $\mathbf{W}_a$ and σ_a^2 can be determined from $\mathbf{S}_a$ in the same way as for a single PPCA model.

A mixture of PPCAs has two advantages over usual mixtures of Gaussian distributions:

- Each component latent variable model locally models both the linear mapping and noise, rather than just the covariance.
- The mixture of PPCA models allows the number of parameters to be controlled by the choice of L. There are fewer parameters per component, compared with $(D+1)D/2$ for a Gaussian with a full covariance matrix.

Although a mixture of PPCA models is appropriate when the data is approximately piece-wise linear, it is still limited by being a locally linear transform. Thus it may not be capable of capturing more complex non-linear structures.

2.5 GTM: the generative topographic mapping

The generative topographic mapping (GTM), developed by Bishop *et al.* (1998), incorporates a non-linear mapping from the latent space to the data space so that it can capture effectively complex correlations in the dataset. The aim then is to represent the high-dimensional data vectors $\{\mathbf{t}_n\}_{n=1,\ldots,N}$ in the latent space so as to reveal important structural characteristics.

The GTM itself is a constrained mixture of Gaussians, since the centres are limited within an L-dimensional non-Euclidean manifold $\mathbf{\Omega}$. It can form a topological mapping because the centres of Gaussians in the data space preserve the structure of the latent space. The parameters of the Gaussian mixture model can be optimized using the EM algorithm.

The GTM provides a principled alternative to the self-organising map (SOM) of Kohonen (1995). The SOM assigns a data point to a single reference vector, whereas the GTM computes the responsibility of all the Gaussians for a data point. Comparing the GTM with the SOM, it is clear that the GTM overcome most of drawbacks suffered by the SOM, such as the lack of probability density model and objective function, the lack of criteria to compare different runs of the SOM procedure, and no general proof of convergence (Bishop *et al.*, 1997).

2.5.1 The GTM model

Imagine that there exists a two-dimensional *rubber sheet* that is embedded in the high-dimensional data space. The GTM covers the cloud of data points by locally stretching, compressing and curving the sheet. The visualisation plot is obtained by first projecting the data points onto the rubber sheet and then letting the rubber sheet "relax" to its original form on the computer screen. We refer to the two-dimensional rubber sheet in the data space as the *projection manifold*, given as follows

$$\mathbf{\Omega} = \{\mathbf{W}\boldsymbol{\phi}(\mathbf{x}) | \mathbf{x} \in \mathcal{H}\}, \tag{2.51}$$

where $\phi(\mathbf{x})$ are M fixed basis functions ϕ_m (including one bias term) and $\mathbf{W}$ is a $D \times M$ matrix. In this way, a non-linear mapping, by using a radial basis functions (RBF) network from points $\mathbf{x}$ in an L-dimensional latent space $\mathcal{H}$ to the data space $\mathcal{D}$ with D-dimension ($L < D$), is defined. For visualisation purposes, the latent space is typically a bounded 2-dimensional Euclidean domain, e.g. $[-1, 1] \times [-1, 1]$. Figure 2.5 illustrates this basic idea of the GTM, where the mapping is given by

$$\mathbf{y}(\mathbf{x}; \mathbf{W}) = \mathbf{W}\phi(\mathbf{x}). \tag{2.52}$$

Since in reality it is impossible that the data can exactly live on the L-dimensional manifold, we need

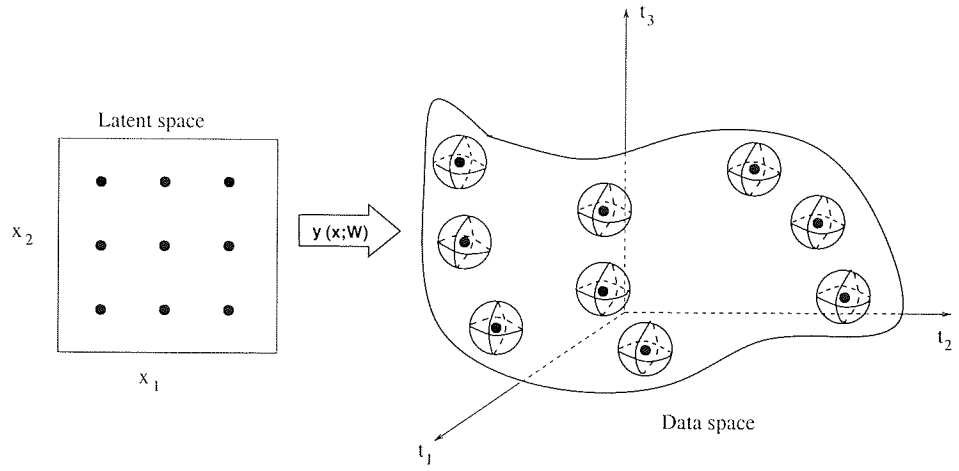


Figure 2.5: GTM mapping and manifold: each node $\mathbf{x}_k$ located at a regular grid in the latent space is mapped to a corresponding point $\mathbf{y}(\mathbf{x}_k; \mathbf{W})$ in the data space, and forms the centre of a corresponding Gaussian distribution.

to define a noise model which is the conditional distribution of data given latent variables. For the GTM, $p(\mathbf{t}|\mathbf{x}, \mathbf{W}, \beta)$ is a spherical Gaussian $\mathcal{N}(\mathbf{y}(\mathbf{x}; \mathbf{W}), \beta^{-1}\mathbf{I})$ centred on $\mathbf{y}(\mathbf{x}; \mathbf{W})$ with variance β^{-1} is used:

$$p(\mathbf{t}|\mathbf{x}, \mathbf{W}, \beta) = (\frac{\beta}{2\pi})^{D/2} \exp\left\{-\frac{\beta}{2}\|\mathbf{t} - \mathbf{y}(\mathbf{x}; \mathbf{W})\|^2\right\}. \tag{2.53}$$

As for the prior distribution $p(\mathbf{x})$ of the latent variable, the second part of specifying a latent variable model, it is a sum of delta functions centred on K nodes $\{\mathbf{x}_k\}_{k=1,\ldots,K}$ of a regular grid in the latent space (which are analogous to the nodes of the SOM (Kohonen, 1995)),

$$p(\mathbf{x}) = \frac{1}{\mathrm{K}} \sum_{k=1}^{\mathrm{K}} \delta(\mathbf{x} - \mathbf{x}_k). \tag{2.54}$$

In this way, a non-Gaussian, maximum entropy (uniform) latent prior is imposed over the latent space. The distribution of data in the data space $\mathcal{D}$, for given values of $\mathbf{W}$ and β, is then obtained by integrating out the latent variables $\mathbf{x}$:

$$p(\mathbf{t}|\mathbf{W}, \beta) = \int p(\mathbf{t}|\mathbf{x}, \mathbf{W}, \beta)p(\mathbf{x})\, d\mathbf{x}. \tag{2.55}$$

From equations (2.54) and (2.55), we have

$$p(\mathbf{t}|\mathbf{W},\beta) = \frac{1}{\mathrm{K}}\sum_{k=1}^{\mathrm{K}} p(\mathbf{t}|\mathbf{x}_k,\mathbf{W},\beta). \tag{2.56}$$

Now we can determine the weight matrix $\mathbf{W}$ and variance β^{-1} using maximum likelihood. The error function is given by the negative log-likelihood of the data:

$$\begin{aligned} E = -\log \mathcal{L}(\mathbf{W},\beta) &= -\log \prod_{n=1}^{N} p(\mathbf{t}_n|\mathbf{W},\beta) \\ &= -\sum_{n=1}^{N} \log\left(\frac{1}{\mathrm{K}}\sum_{k=1}^{K} p(\mathbf{t}_n|\mathbf{x}_k,\mathbf{W},\beta)\right). \end{aligned} \tag{2.57}$$

Since the GTM model involves a Gaussian mixture model, it is possible to train the model by using the EM algorithm.

- $E-$step

  To compute the responsibilities R_{kn} given by:

$$\begin{aligned} R_{kn} &= p(\mathbf{x}_k|\mathbf{t}_n,\mathbf{W},\beta) \\ &= \frac{p(\mathbf{t}_n|\mathbf{x}_k,\mathbf{W},\beta)p(\mathbf{x}_k)}{\sum_{k'=1}^{K} p(\mathbf{t}_n|\mathbf{x}_{k'},\mathbf{W},\beta)p(\mathbf{x}_{k'})}. \end{aligned} \tag{2.58}$$

  R_{kn} corresponds to the posterior probability that the nth data point was generated by the kth component.

- $M-$step

  Responsibilities calculated in $E-$step will act as weights in the update equations for $\mathbf{W}$ and β.

  - A matrix equation for $\mathbf{W}$ is given by:

$$\mathbf{W}\mathbf{\Phi}\mathbf{G}\mathbf{\Phi}^T = \mathbf{T}\boldsymbol{R}^T\mathbf{\Phi}^T, \tag{2.59}$$

    where $\mathbf{T}$ is the $D \times N$ matrix containing the data points, $\mathbf{R}$ is the $K \times N$ responsibility matrix with elements defined in equation (2.58), $\mathbf{\Phi}$ is the $M \times K$ RBF matrix with elements $\Phi_{mk} = \phi_m(\mathbf{x}_k)$ and $\mathbf{G}$ is an $K \times K$ diagonal matrix with entries

$$G_{kk} = \sum_{n=1}^{N} R_{kn}. \tag{2.60}$$

  - A re-estimation formula for β is given by

$$\frac{1}{\beta} = \frac{1}{ND}\sum_{n=1}^{N}\sum_{k=1}^{K} R_{kn}\|\mathbf{y}(\mathbf{x_k};\mathbf{W}^{new}) - \mathbf{t}_n\|^2, \tag{2.61}$$

  Where $\mathbf{W}^{new}$ corresponds to the updated weights from equation (2.59), which means that we must first minimise with respect to the weights, then with respect to β.

2.5.2 Parameter selection

1. Typically, a Gaussian RBF network is used for the mapping. The ratio of their width parameter σ to the spacing of basis functions controls the smoothness of the manifold (see equation (2.51)) in the data space.

   When the RBF has a large number of degrees of freedom, the map $\mathbf{y}(\mathbf{x}; \mathbf{W})$ may be very complex and the manifold may has regions of large curvature. In order to solve this problem, a weight decay regularisation term λ can be introduced to control the smoothness properties of the mapping function (2.51). This leads to a modification to the $M-$step (2.59) to give

   $$\mathbf{W}(\mathbf{\Phi G \Phi}^T + \lambda \mathbf{I}) = \mathbf{T} \boldsymbol{R}^T \mathbf{\Phi}^T, \tag{2.62}$$

   where $\mathbf{I}$ is the identity matrix.

2. Here we must indicate that the number K of latent grid points $\mathbf{x}_k$ can sometimes be more than the number N of data points. This is because the number of degrees of freedom in the GTM is controlled by the basis functions, which is independent of the number of latent grid nodes. In practice, however, using a dense grid is computationally prohibitive.

3. We need an initialisation method which is fast to compute and reasonably close to the optimal solution. A linear method can satisfy the speed requirement. We use PCA to initialise the parameters $\mathbf{W}$. An L-dimensional linear subspace is used as initial manifold. We compute the data covariance matrix and obtain the first and second principal eigenvectors, and then we determine $\mathbf{W}$ by minimising the sum-of-squares error function

   $$\mathrm{E} = \frac{1}{2} \sum_k \| \mathbf{W} \boldsymbol{\phi}(\mathbf{x_k}) - \mathbf{U} \mathbf{x_k} \|^2, \tag{2.63}$$

   where the columns of $\mathbf{U}$ are given by the eigenvectors. The value of β^{-1} is initialised to be a larger value, the minimum of the $(L+1)$st eigenvalues from PCA and half the average squared distance between centres of Gaussian mixture, to prevent premature convergence of the main EM algorithm (Nabney, 2001).

2.5.3 Data visualisation

The GTM provides a full posterior distribution R_{kn} as a result of the Bayesian approach. This distribution can be very difficult to visualise. To see all the projections of the data points at once in a single plot, one can summarise the posterior by its mean, given for each data point $\mathbf{t}_n$ by

$$\langle \mathbf{x} | \mathbf{t_n}, \mathbf{W}, \beta \rangle = \int p(\mathbf{x} | \mathbf{t}_n, \mathbf{W}, \beta) \mathbf{x} \, d\mathbf{x}. \tag{2.64}$$

With the choice of prior distribution (2.54), we have

$$\langle \mathbf{x} | \mathbf{t_n}, \mathbf{W}, \beta \rangle = \sum_{k=1}^{K} R_{kn} \mathbf{x}_k. \tag{2.65}$$

An alternative approach is to evaluate the mode of the distribution, given by

$$k^{max} = \underset{\{k\}}{\text{argmax}}\, R_{kn}. \tag{2.66}$$

A visualisation plot of HTS data with a GTM is shown in Figure 2.6. There is a white band along the middle left of the figure, which separates most inactive compounds from the other three classes. Looking at the left part of the figure, we can find two clusters including mixed compounds with a different number of activities, which are parted from most of the inactive compounds. It suggests that the structure captured with the GTM is much more clearly represented than with simple PCA approach (see Figure 2.4) and the projections are well separated.

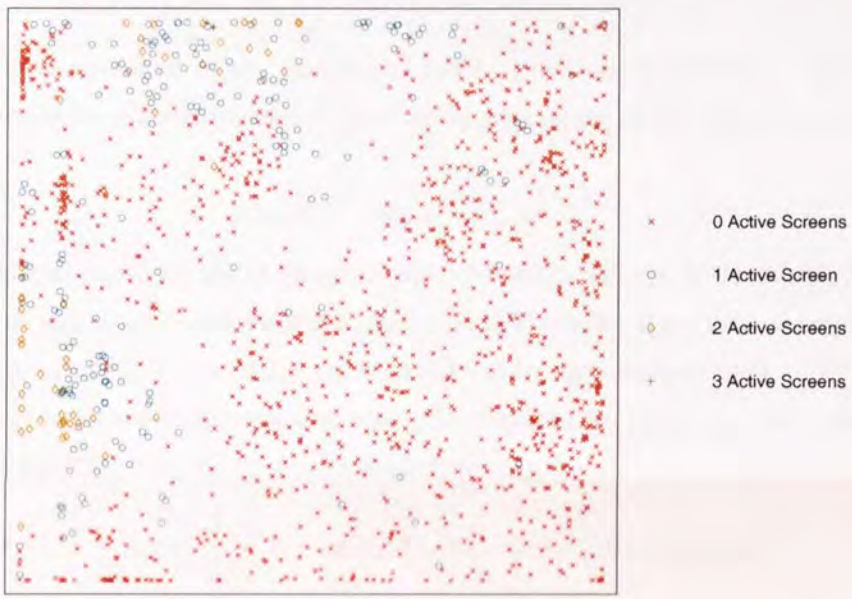


Figure 2.6: Projection of the HTS data using the GTM: projections of data points were computed from posterior mean (equation (2.65)). The latent space is bounded in a 2-dimensional Euclidean domain, i.e. $[-1, 1] \times [-1, 1]$.

2.6 Latent trait models

2.6.1 The latent trait model

In the statistics literature, latent trait models are used when the observed variables are categorical, while the latent variables are continuous (see Table 1.1). However, this was extended in (Moustaki, 1996) for mixed (discrete and continuous) observed variables. In this thesis, we use the term "trait" to distinguish continuous latent variables from categorical, while the observed data may be either discrete or continuous; the same terminology was used in (Kabán and Girolami, 2001).

Since the latent trait model is generative, it provides powerful and principled tools of data analysis and visualisation in terms of topographic organization. The latent trait model developed by Kabán and Girolami (2001) is a generalised version of the GTM. It allows us to deal with both continuous and discrete observed variables in a unified framework.

Consider a trait model $\mathcal{M}$. The system's noise is modeled in a parametric form as a member of the exponential family of distributions. It may be either the widely utilized Gaussian model (as in the GTM), which is appropriate in cases of real valued continuous observations, or discrete distributions, such as the binomial, the multinomial and Poisson distributional models. The parameterised functional form ((Amari, 1985), (Barndorff-Nielsen, 1978)) of a distribution from the exponential family is the following,

$$p_{\mathcal{G}}(\mathbf{t}_n|\mathbf{x}_k,\boldsymbol{\theta}_{\mathcal{M}}) = \exp\left\{\mathbf{y}(\mathbf{x}_k;\boldsymbol{\theta}_{\mathcal{M}})\mathbf{t}_n - \mathcal{G}(\mathbf{y}(\mathbf{x}_k;\boldsymbol{\theta}_{\mathcal{M}}))\right\} p_0(\mathbf{t}_n), \tag{2.67}$$

where $\mathcal{G}(\cdot)$ denotes the cumulant function given by

$$\mathcal{G}(\mathbf{y}(\mathbf{x};\boldsymbol{\theta}_{\mathcal{M}})) = \log\left(\int \exp\left(\mathbf{y}(\mathbf{x};\boldsymbol{\theta}_{\mathcal{M}})\mathbf{t}\right)p_0(\mathbf{t}))\ d\mathbf{t}\right), \tag{2.68}$$

and $p_0(\mathbf{t}_n)$ is a factor independent of the parameter. Further, the non-linearity $\mathbf{y}(\cdot)$, which is also called the natural parameter of the exponential distribution in equation (2.67), is conveniently chosen of the form

$$\mathbf{y}(\mathbf{x}_k;\boldsymbol{\theta}_{\mathcal{M}}) = \boldsymbol{\theta}_{\mathcal{M}}\boldsymbol{\phi}(\mathbf{x}_k), \tag{2.69}$$

where $\boldsymbol{\theta}_{\mathcal{M}}$ is a $D \times M$ parameter matrix of the trait model $\mathcal{M}$ and $\boldsymbol{\phi}(\mathbf{x}_k)$ are M fixed basis functions. These could be any smooth functions; typically Gaussian radial basis functions may be employed and also a bias function is included. The notation $\boldsymbol{\phi}_k = \boldsymbol{\phi}(\mathbf{x}_k)$ will be used as shorthand.

Similar to the GTM, the latent space is a regular grid of points $\{\mathbf{x}_k\}_{k=1,\ldots,K}$. The prior $p(\mathbf{x})$ is given by equation (2.54). The data log-likelihood is the following,

$$\log\mathcal{L}(\boldsymbol{\theta}_{\mathcal{M}}) = \sum_{n=1}^{N}\log\frac{1}{K}\left\{\sum_{k=1}^{K}\exp\left\{\mathbf{y}(\mathbf{x}_k;\boldsymbol{\theta}_{\mathcal{M}})\mathbf{t}_n - \mathcal{G}(\mathbf{y}(\mathbf{x}_k;\boldsymbol{\theta}_{\mathcal{M}}))\right\}p_0(\mathbf{t}_n)\right\}. \tag{2.70}$$

To train the model using the EM algorithm, the expectation of the complete data log-likelihood E_{comp} of the model is computed, which in this case is written as follows.

$$\langle E_{comp}\rangle = \sum_{n=1}^{N}\sum_{k=1}^{K}R_{kn}\left\{\mathbf{y}(\mathbf{x}_k;\boldsymbol{\theta}_{\mathcal{M}})\mathbf{t}_n - \mathcal{G}(\mathbf{y}(\mathbf{x}_k;\boldsymbol{\theta}_{\mathcal{M}})) + \log p_0(\mathbf{t}_n) + \log\frac{1}{K}\right\}, \tag{2.71}$$

where R_{kn} is computed via Bayes' theorem in the $E-$step:

$$R_{kn} = \frac{p_{\mathcal{G}}(\mathbf{t}_n|\mathbf{x}_k,\boldsymbol{\theta}_{\mathcal{M}})}{\sum_{k'=1}^{K}p_{\mathcal{G}}(\mathbf{t}_n|\mathbf{x}_{k'},\boldsymbol{\theta}_{\mathcal{M}})}, \tag{2.72}$$

referred to as *responsibilities* of the latent point $\mathbf{x}_k$ in having generated $\mathbf{t}_n$.

Taking the derivative of $\langle E_{comp}\rangle$ with respect to the parameter $\boldsymbol{\theta}_{\mathcal{M}}$, we obtain

$$\frac{\partial\langle E_{comp}\rangle}{\partial\boldsymbol{\theta}_{\mathcal{M}}} = \left\{\mathbf{T}\boldsymbol{R}^T - \mathbf{g}(\boldsymbol{\theta}_{\mathcal{M}}\boldsymbol{\Phi})\boldsymbol{G}\right\}\boldsymbol{\Phi}^T. \tag{2.73}$$

$\boldsymbol{R}$ is the $K \times N$ responsibility matrix with elements R_{kn} and $\boldsymbol{G}$ is a $K \times K$ diagonal matrix with elements $G_{kk} = \sum_{n=1}^{N}R_{kn}$, $\boldsymbol{\Phi}$ is an $M \times K$ RBF matrix with $\boldsymbol{\phi}_k$ in its k-th column and $\mathbf{T}$ is the data matrix. The function $\mathbf{g}(\cdot)$ denotes the gradient of the cumulant function $\mathcal{G}(\cdot)$ and is termed as the *inverse link* function (McCullagh and Nelder, 1985) of the distribution, as it makes the link between the natural parameter $\mathbf{y}(\cdot)$ and the expectation parameter of the distribution. That is

$$\langle\mathbf{t}|\mathbf{x}_k\rangle := \mathbf{g}(\boldsymbol{\theta}_{\mathcal{M}}\boldsymbol{\phi}_k) = \nabla_{\boldsymbol{\theta}_{\mathcal{M}}\boldsymbol{\phi}_k}\mathcal{G}(\boldsymbol{\theta}_{\mathcal{M}}\boldsymbol{\phi}_k)), \tag{2.74}$$

where ∇ denotes the gradient operator.

An example is a normal distribution with mean μ and unit variance, which has a density that is

$$p(\mathbf{t}|\mu) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}(\mathbf{t}-\mu)^2\right\}.$$

It may be rewritten as the form of the equation (2.67)

$$p(\mathbf{t}|\mu) = \exp\left\{\mu\mathbf{t} - \frac{\mu^2}{2}\right\} p_o(\mathbf{t})$$

with $\mathbf{y}(\cdot) = \mu$, $\mathcal{G}(\cdot) = \frac{\mu^2}{2}$ and $\log p_0(\mathbf{t}) = -\log(\sqrt{2\pi}) - \frac{\mathbf{t}^2}{2}$. Then we have $\mathbf{g}(\cdot) = \mu$. It is clear that $\mathbf{g}(\cdot)$ is an identity. In this case, by setting the derivative of $\langle E_{comp}\rangle$ equal to zero, one obtains the closed form $M-$step of the GTM, given by

$$\mathbf{T}\boldsymbol{R}^T\boldsymbol{\Phi}^T = \theta_{\mathcal{M}}\boldsymbol{\Phi}\boldsymbol{G}\boldsymbol{\Phi}^T.$$

As for the case in which the Gaussians are isotropic, the variance β^{-1} must be calculated by equation (2.61).

In general, however, a non-linear optimization technique may be required. In practice, we used the following gradient-based update (Kabán and Girolami, 2001) in the $M-$step:

$$\boldsymbol{\theta}_{\mathcal{M}}^{new}\boldsymbol{\phi}_k = \boldsymbol{\theta}_{\mathcal{M}}^{old}\boldsymbol{\phi}_k + \eta\sum_{n=1}^{N}\sum_{k'=1}^{K}(\mathbf{t}_n - \langle\mathbf{t}|\mathbf{x}_{k'}\rangle)R_{k'n}\boldsymbol{\phi}_{k'}^T\boldsymbol{\phi}_k. \tag{2.75}$$

After training, the latent space representation of the point $\mathbf{t}_n$ is taken to be the posterior mean $\langle\mathbf{x}|\mathbf{t}_n\rangle$, which, according to the discretisation, is computed simply as $\sum_{k=1}^{K} R_{kn}\mathbf{x}_k$.

The experimental result

As an example, we did an experiment on a small text dataset containing 387 instances coded as 100-dimensional binary vectors. The dataset is grouped into 4 topics. The tool used for pre-processing the document data is the same as the one mentioned in section 1.5.4. The latent space is bounded in a 2-dimensional Euclidean domain, i.e. $[-1,1]\times[-1,1]$.

First we trained the LTM with Bernoulli distribution. The result is present in Figure 2.7. It shows that the 4 classes are separated very well and the model effectively captured the structure in the dataset. We also applied the GTM model for the same dataset. The result is displayed in Figure 2.8, where some projections of data points from the different classes are mixed and many points in the same class are overlapped. In comparison with the LTM, it suggests that the GTM is a less effective choice when fitting the discrete dataset.

2.6.2 Local magnification factors of the latent trait manifolds

The term "magnification factor" refers to the degree of stretching or compression of the latent space when embedded into the data space. Consider the Cartesian coordinate system in the latent space and the mapping of this space to a curvilinear coordinate system defined in the manifold embedded

Figure 2.7: Projection of document data using the LTM: the noise model is Bernoulli distribution.

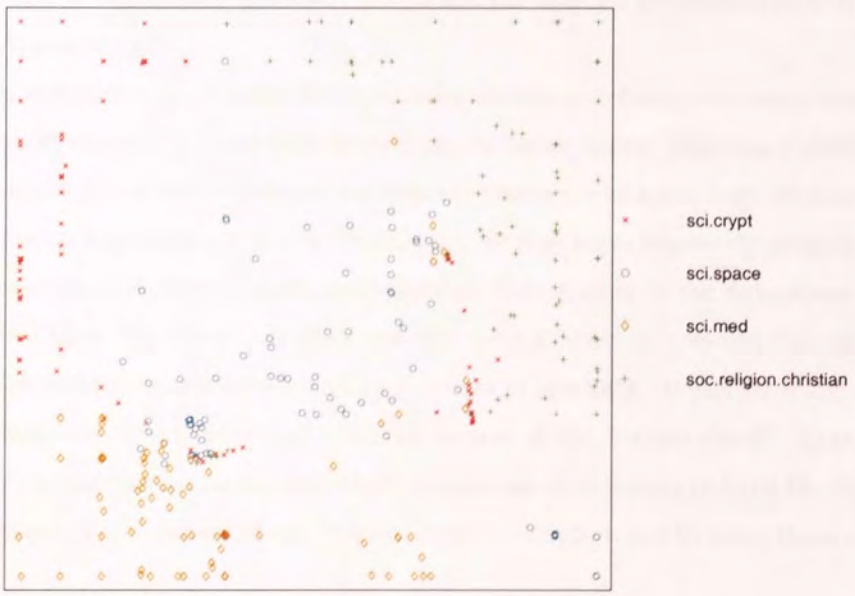


Figure 2.8: Projection of document data using the GTM: the noise model is spherical Gaussian distribution.

in the data space. It has been shown in (Bishop *et al.*, 1997) that the local magnification factor corresponding to a point $\mathbf{x}_0$ in the latent space, defined as the ratio between the area in the projection manifold Ω and the corresponding area of an infinitesimal rectangle in the latent Cartesian space, is given by $\sqrt{|\boldsymbol{S}(\mathbf{x}_0)|}$, where $|\boldsymbol{S}(\mathbf{x}_0)|$ denotes the determinant of the metric tensor $\boldsymbol{S} = \boldsymbol{\Upsilon}^T\boldsymbol{\Upsilon}$, in which case $\boldsymbol{\Upsilon}$ denotes the Jacobian of the mapping from the latent space to the data space. For the GTM, $\boldsymbol{\Upsilon}$ is the Jacobian of the mapping (2.51).

A general latent trait model defines a density in the data space, using a smooth mapping from the latent space to the data space,

$$\Omega: \quad \mathcal{H} \to \mathcal{D}, \quad \Omega(\mathbf{x}) = g(\boldsymbol{\theta}_{\mathcal{M}}\boldsymbol{\Phi}(\mathbf{x})). \tag{2.76}$$

We refer to the manifold $\Omega(\mathcal{H})$ as the *projection manifold* of the LTM. So the Jacobian of the mapping is given by

$$\boldsymbol{\Upsilon} = \frac{\partial \mathbf{g}(\boldsymbol{\theta}_{\mathcal{M}}\boldsymbol{\phi}(\mathbf{x}_0))}{\partial \mathbf{x}} = \boldsymbol{\mathcal{I}}\boldsymbol{\theta}_{\mathcal{M}}\boldsymbol{V}, \tag{2.77}$$

where the $M \times K$ matrix $\boldsymbol{V}$ is equal to $\left(\frac{\partial \phi_m(\mathbf{x})}{\partial \mathbf{x}_k}|_{\mathbf{x}=\mathbf{x}_0}\right)_{m=1,\ldots,M,k=1,\ldots,K}$, and the $D \times D$ matrix $\boldsymbol{\mathcal{I}} = \left(\frac{\partial g_{d'}(\boldsymbol{y})}{\partial y_d}|_{\boldsymbol{y}=\boldsymbol{\theta}_{\mathcal{M}}\boldsymbol{\phi}(\mathbf{x}_0)}\right)_{d'=1,\ldots,D,d=1,\ldots,D}$ is the Fisher information matrix of the noise distribution. If Gaussian radial basis functions are utilized for $\phi(\cdot)$, then the (m,k)-th element of the matrix $\boldsymbol{V}$ will be $v_{m,k} = -\phi_m(\mathbf{x}_0)(\mathbf{x}_k - c_{m,k})\sigma^{-2}$ where $c_{m,k}$ denotes the k-th coordinate of the radial basis centre which corresponds to the m-th basis function.

Thus the magnification factor associated with a point $\mathbf{x}_0$ in the latent space is $\sqrt{|\boldsymbol{V}^T\boldsymbol{\theta}_{\mathcal{M}}^T\boldsymbol{\mathcal{I}}^T\boldsymbol{\mathcal{I}}\boldsymbol{\theta}_{\mathcal{M}}\boldsymbol{V}|}$.

In the case of Gaussian noise models, the matrix $\boldsymbol{\mathcal{I}}^T\boldsymbol{\mathcal{I}}$ is the identity matrix. Note also that in all independent noise models this matrix will be diagonal; therefore the increase in computational complexity will not be significant. However, this is not the case for the multinomial trait model (as can be seen in Appendix B.3).

The absolute values of magnification factors can be viewed as defining how many times the area is magnified when it is mapped into the data space from the latent space. They are a useful tool in data visualisation by highlighting the boundaries between the clusters, which has been illustrated in (Bishop *et al.*, 1997) for several experimental results. Practically, we may superimpose the magnification factors on the latent space visualisation to understand the data distribution in the data space. An example can be viewed in Figure 2.9, where the data was the same as used in previous experiment. We see that the highly stretched regions correspond to 4 classes of the data. It can be imagined that data points in each class were mostly projected onto the corners of the "rubber sheet". Since in each class there are fewer data points comparing with their dimensions, it is somehow hard for the manifold to try fitting these data. Thus corners of the "rubber sheet" were stretched to cover those corresponding data points.

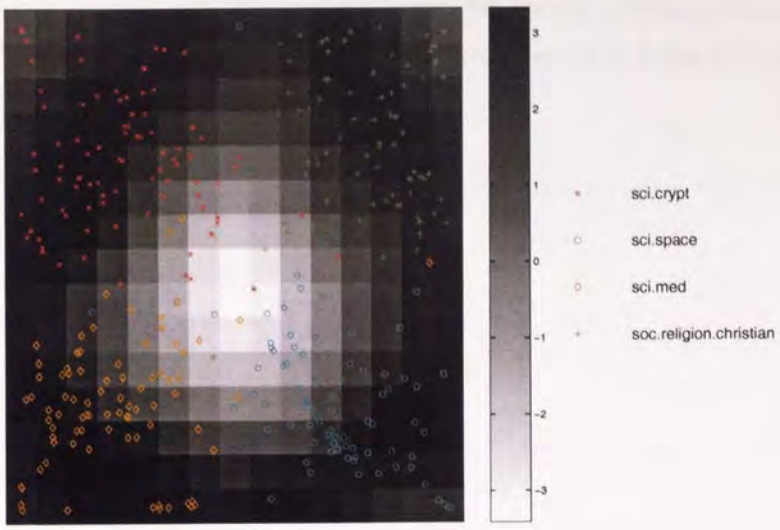


Figure 2.9: Projection of document data using the LTM with magnification factors, scaled by $\log_2$.

2.7 Neuroscale

Neuroscale is related to Sammon Mapping (Sammon, 1969) and Multidimensional Scaling (Kruskal, 1964), which are classical projection visualisation methods. It uses radial basis function networks to predict the coordinates of the data point in the transformed feature space (Lowe and Tipping, 1997).

Using Neuroscale, the geometric structure of the data is optimally preserved in the transformation, and the embodiment of this constraint is that the inter-point distances in the feature space should match as closely as possible to the corresponding distances in the data space.

Usually, the inter-point distances are measured with a Euclidean metric:

$$d^*_{nn'} = \| \mathbf{t}_n - \mathbf{t}_{n'} \| . \tag{2.78}$$

Each data point $\mathbf{t}_n$ in $\mathcal{R}^D$ is transformed to a corresponding point $\mathbf{y}_n$ in the feature space $\mathcal{R}^L$. The error of the mapping is measured by *stress metric*,

$$\mathrm{E} = \sum_{n}^{\mathrm{N}} \sum_{n'>n}^{\mathrm{N}} (d^*_{nn'} - d_{nn'})^2, \tag{2.79}$$

where $d_{nn'} = \| \mathbf{y}_n - \mathbf{y}_{n'} \|$.

The points $\mathbf{y}_n$ are predicted by a radial basis function neural network, maybe written as:

$$\mathbf{Y} = \mathbf{\Phi W}, \tag{2.80}$$

where $\mathbf{Y}$ is a matrix of row feature vectors $\mathbf{y}_n$, $\mathbf{W}$ is the $M \times L$ weight matrix and $\mathbf{\Phi}$ is the $N \times M$ basis functions matrix.

2.7.1 The shadow-targets algorithm

There are a number of standard optimisation techniques for non-linear problems, such as *scaled conjugate gradient, on-line gradient descent*, and so on, which may be applied to minimisation of *stress*

metric in topographic mapping contexts. Rather than using general purpose optimisation algorithms, Tipping and Lowe (1997) proposed *shadow-targets* algorithm, which is faster in converging.
The equations for calculating the weight derivatives are:

$$\frac{\partial \mathrm{E}}{\partial w_{ml}} = \sum_{n}^{\mathrm{N}} \frac{\partial \mathrm{E}}{\partial \mathbf{y}_n} \frac{\partial \mathbf{y}_n}{\partial w_{ml}}, \tag{2.81}$$

where

$$\frac{\partial \mathrm{E}}{\partial \mathbf{y}_n} = -2 \sum_{n' \neq n} \left(\frac{d^*_{nn'} - d_{nn'}}{d_{nn'}} \right) (\mathbf{y}_n - \mathbf{y}_{n'}). \tag{2.82}$$

In a supervised problem, the error E is given by

$$\mathrm{E} = \frac{1}{2} \sum_{n=1}^{N} \| \mathbf{y}_n - \mathbf{x}_n \|^2, \tag{2.83}$$

where the vectors $\mathbf{x}_n$ are the explicit targets, such that

$$\frac{\partial \mathrm{E}}{\partial \mathbf{y}_n} = (\mathbf{y}_n - \mathbf{x}_n). \tag{2.84}$$

Combining equation (2.82) with equation (2.84) gives a set of vectors that can be considered to represent (or shadow) estimated targets $\hat{\mathbf{x}}_n$:

$$\hat{\mathbf{x}}_n = \mathbf{y}_n - \frac{\partial \mathrm{E}}{\partial \mathbf{y}_n} \tag{2.85}$$

$$= \mathbf{y}_n + 2 \sum_{n' \neq n} \left(\frac{d^*_{nn'} - d_{nn'}}{d_{nn'}} \right) (\mathbf{y}_n - \mathbf{y}_{n'}). \tag{2.86}$$

For a fixed set of estimated targets $\hat{\mathbf{x}}_n$, the least squares can be solved directly by

$$\mathbf{W} = \mathbf{\Phi}^{\dagger} \hat{\mathbf{X}}, \tag{2.87}$$

where $\mathbf{\Phi}^{\dagger} = \left[\mathbf{\Phi}^T \mathbf{\Phi} \right]^{-1} \mathbf{\Phi}^T$ is the pseudo-inverse of $\mathbf{\Phi}$ (see e.g. (Horn and Johnson, 1985)).

But $\hat{\mathbf{x}}_n$ are not fixed due to the change of $\frac{\partial \mathrm{E}}{\partial \mathbf{y}_n}$. Thus an approach is to repetitively estimate the targets at each iteration. However, the targets estimated by (2.86) may be poor in the early training stage, and hence lead to the increase of the error. Usually a parameter η set to a value in the range $(0, 1)$ can be introduced to cope with this problem. To estimate the targets, we have

$$\hat{\mathbf{x}}_n = \mathbf{y}_n - \eta \frac{\partial \mathrm{E}}{\partial \mathbf{y}_n}. \tag{2.88}$$

η is initially small, then is increased as E decreases during the training procedure. The algorithm is referred to as the *shadow-targets* algorithm, because it is *shadowing* the standard Sammon Mapping generation procedure where a gradient-descent optimisation is used, when the estimated target $\hat{\mathbf{x}}_n$ is identical to the new point by using the equation (2.88).

The experimental result

First we visualised the oil dataset with Neuroscale; the result is shown in Figure 2.10. It is similar to the result presented in Figure 2.3. However, *Annular* and *Laminar* classes do not appear so tightly

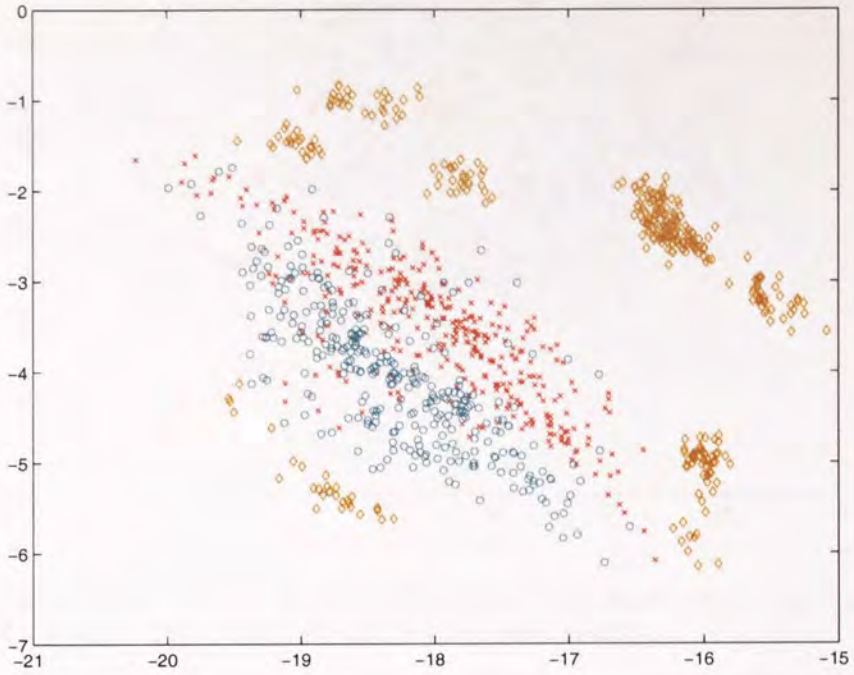


Figure 2.10: Visualisation of the oil data with Neuroscale. Three classes are shown: Homogeneous (crosses), Annular (circles) and Laminar (diamond signs).

grouped as those obtained using PCA. In addition, the *Laminar* class is grouped in 7 distinct clusters rather than 5.

Figure 2.11 presents the projection of the HTS data using Neuroscale. Looking at the plot, we see that Neuroscale can separate most of the inactive compounds from other classes. This is an improvement in comparison with using PCA on the same dataset (see Figure 2.4), because Neuroscale uses a global non-linear method rather than a linear projection. However, Neuroscale is not yet an ideal way for visualising this dataset since there is no obvious border between clusters and many points are overlapped.

2.8 Principal curves and surfaces

In this section we shall briefly mention a recently developed extension to PCA, namely principal curves and surfaces, since they are another important non-linear model used for visualisation.

Principal curves defined by Hastie and Stuetzle (1989) is a non-linear generalisation of principal components analysis. It provides a smooth one-dimensional curved approximation, passing through the "middle" of the data cloud in the data space, such that each point of the curve is the average of all data points that project onto it. A principal surface provides a curved manifold approximation of dimension 2 or more, and its algorithm is computationally more demanding. Practically, the curves must be discretised to be computed. Ritter *et al.* (1992) gives a view that discretised principal curves are essentially equivalent to Self-Organizing Maps.

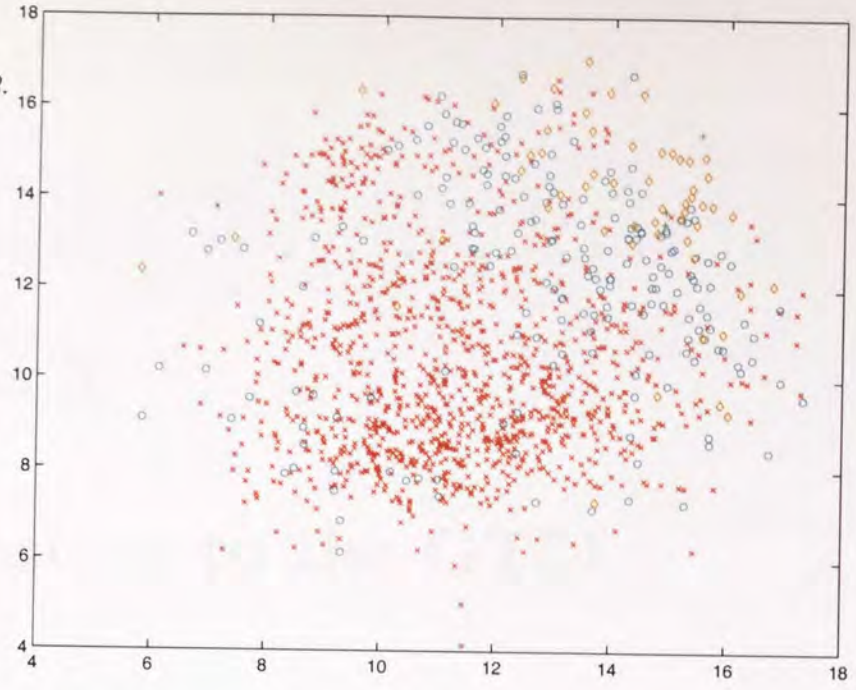


Figure 2.11: Visualisation of the HTS data with Neuroscale. Four classes are shown: *0-active* (crosses), *1-active* (circles), *2-active* (diamond signs) and *3-active* (plus signs).

Tibshirani (1992) defines a generative variant of the principal curve model based on a mixture model, which can be trained using the EM algorithm. Recently, Kegl *et al.* (2000) develop principal curves as continuous curves of a given length which minimise the expected squared distance between the curve and points of the space according to a given distribution. However, as discussed in (Svensén, 1998), when compared to various principal curve and surface models, latent variable models like GTM provide an alternative which has attractive computational advantages when modelling two or three-dimensional distributions, and can also be employed with higher dimensional latent space.

2.9 Discussion

In this chapter we have reviewed some specified latent variable models including probabilistic PCA, the GTM, the LTM and Neuroscale based on data topography as well. To compare these algorithms, we did experiments on the HTS dataset. Results suggest that the GTM, a special case of the LTM, will be a more suitable tool to further study for our visualisation task, whereas PCA based on the simple linear transformation, by contrast, cannot successfully capture complex intrinsic structures of the data. Structures in this dataset captured by Neuroscale is better than with simple PCA, but not so distinct as with the GTM. However, we must indicate that Neuroscal has its own abilities and advantages in non-linear mapping. In latter chapters, we will focus on further developing both the GTM and the LTM .

Chapter 3

Extensions to the GTM

In this chapter we discuss two novel extensions to the GTM. **(1)** Local directional curvatures. Since the GTM forms a "smooth" two-dimensional projection manifold, we can analytically compute its local directional curvature at any point on the manifold using tools from differential geometry. Curvature plots are useful for discovering regions where the projection manifold is bent. **(2)** Visualisation of the dataset involving missing values using class information constraints (that is data with class labels). This is helpful in the construction of informative visualisation plots, even when many of the training points are incomplete.

3.1 Introduction

With the aid of projection visualisation plots, it is possible to see relationships among data points, learn information about clusters and find "outliers" in the high dimensional data space. However, a visualisation plot by itself is usually not enough when the mapping from the latent space to the data space is non-linear. We need a set of tools for monitoring the "amount of non-linearity" in the projection manifolds. These knowledge can only be obtained from the *geometry* of the projection manifold in the high dimensional data space.

Magnification factors, which are discussed in section 2.6.2, describe how regions in the latent (visualisation) space are stretched or compressed when mapped to the data space. For example, regions of stretch in the manifold can highlight boundaries between well-separated clusters in the data space. Magnification factors can represent the extent to which the regions are magnified on projection to the data space, however, it may be rather difficult to interpret if the manifold has a complex shape in the data space. So the user may also need second-order quantities, such as local curvature, which quantifies the bending of complicated folds in the data space. Since the GTM forms a "smooth" two-dimensional projection manifold, we can analytically compute its local directional curvature at any point on the manifold using tools from differential geometry.

When visualising data points, one is often faced with the problem of incomplete data. Often, this

is solved via the deletion of incomplete records. However, this commonly results in the loss of useful information. For example, in drug design, data with missing values may include one or more new drug candidates. Therefore it is important to use all available values as well as any additional information to reconstruct or infer the missing values. It is often the case that the visualisation plot is more helpful when data is labelled with class information (using color or sign for each class, for example), which is readily available in many datasets. Here, we would like to find a way of training the GTM model with incomplete data, and also reconstruct the missing values using additional class information. In this way the data, including the missing components, can be shown in a visualisation plot that is as "faithful" as possible.

Formulation of the local curvatures of the GTM manifold is detailed in section 3.2. Section 3.3 gives a detailed description of the process of incorporating class information into the GTM training process. The usefulness of these extensions are further illustrated with experimental results, which are presented in the respective sections.

3.2 Local directional curvatures of GTM manifolds

3.2.1 Derivation

For the GTM, given a point $\mathbf{x} \in \mathcal{H}$ in the latent space, its image under the map $\mathbf{y}$ is

$$\mathbf{y}(\mathbf{x}) = \mathbf{W}\boldsymbol{\phi}(\mathbf{x}), \tag{3.1}$$

where $\mathbf{W}$ is a $D \times M$ matrix of weight parameters and $\boldsymbol{\phi}(\mathbf{x}) = (\phi_0(\mathbf{x}), \ldots, \phi_M(\mathbf{x}))^T$ including a bias term.

The image of the latent space $\mathcal{H}$ forms a *smooth* L-dimensional manifold in the data space, which is referred as the projection manifold $\boldsymbol{\Omega}$, given by

$$\boldsymbol{\Omega} = \mathbf{y}(\mathcal{H}) = \{\mathbf{y}(\mathbf{x}) \in \mathcal{R}^D | \mathbf{x} \in \mathcal{H}\}. \tag{3.2}$$

Now imagine a simple case in which $y(x)$ is one-dimensional. The first derivative of $y(x)$ gives the slope of the curve, while the second derivative gives the rate of change of slope, which is related to the curvature. Generally, the local curvature of $y(x)$ can be described by two orthogonal components, one being normal to the slope and the other tangential, involving second order partial derivatives. The normal component measures how much the image is curved as x changes. The tangential component measures the change of the projection in the direction of the slope. Our aim is to calculate the normal component of the second derivative.

Figure 3.1 illustrates the basic idea of directional curvature in the case of a two-dimensional latent space.

Let $\mathbf{h}$ be a unit directional vector, $\mathbf{h} = (h_1, h_2, ..., h_L)^T$. Let $\mathbf{x}(b)$, in the latent space $\mathcal{H}$, represent a straight line passing through a point $\mathbf{x}_0 (\mathbf{x}_0 \in \mathcal{H})$ along $\mathbf{h}$:

$$\mathbf{x}(b) = \mathbf{x}_0 + b\mathbf{h}, \tag{3.3}$$

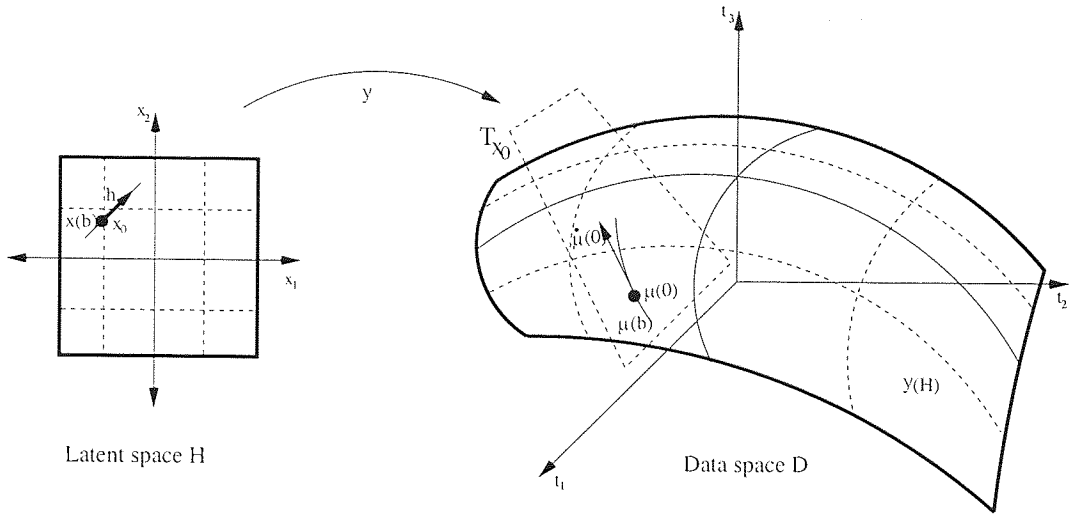


Figure 3.1: Explanation of local directional derivative of the visualisation manifold. A straight line $\mathbf{x}(b)$ passing through the point $\mathbf{x}_0$ in the latent space $\mathcal{H}$ is mapped via $\mathbf{y}$ to the curve (lifted line) $\mu(b) = \mathbf{y}(\mathbf{x}(b))$ in the data space $\mathcal{D}$. Curvature of μ at $\mathbf{y}(\mathbf{x}_0) = \mu(0)$ is related to the directional curvature of the projection manifold $\mathbf{y}(\mathcal{H})$ with respect to the direction $\mathbf{h}$. The tangent vector $\dot{\mu}(0)$ to μ at $\mu(0)$ lies in $\mathbf{T}_{\mathbf{x}_0}$ (dashed rectangle), the tangent plane of the manifold $\mathbf{y}(\mathcal{H})$ at $\mu(0)$.

which is defined by the parameters $b \in \mathcal{R}$.

As the parameter b varies, the image of the line $\mathbf{x}(b)$ generates a curve in the projection manifold $\mathbf{\Omega}$,

$$\mu(b) = \mathbf{y}(\mathbf{x}_0 + b\mathbf{h}), \tag{3.4}$$

called a lifted line. The tangent to this curve at $\mathbf{y}(\mathbf{x}_0) = \mu(0)$ is

$$\begin{aligned} \dot{\mu}(0) &= \left[\frac{\mathrm{d}\mu(b)}{\mathrm{d}b}\right]_{b=0} \\ &= \left[\sum_{r=1}^{L} \frac{\partial \mathbf{y}(\mathbf{x})}{\partial x_r} \frac{\mathrm{d}x_r(b)}{\mathrm{d}b}\right]_{\mathbf{x}=\mathbf{x}_0, b=0} \\ &= \sum_{r=1}^{L} \mathbf{\Gamma}_r^{(1)} h_r \\ &= \mathbf{\Gamma}^{(1)} \mathbf{h}, \end{aligned} \tag{3.5}$$

where

$$\begin{aligned} \mathbf{\Gamma}_r^{(1)} &= \mathbf{W}\Psi_r^{(1)}(\mathbf{x}_0) \\ &= \mathbf{W}\left(\frac{\partial \phi_1(\mathbf{x}_0)}{\partial x_r}, \frac{\partial \phi_2(\mathbf{x}_0)}{\partial x_r}, \ldots, \frac{\partial \phi_M(\mathbf{x}_0)}{\partial x_r}\right)^T \end{aligned} \tag{3.6}$$

is a (column) vector of partial derivatives of the GTM map $\mathbf{y}$ (at $\mathbf{x}_0 \in \mathcal{H}$) with respect to the r-th latent space variable x_r, and $\mathbf{\Gamma}^{(1)}$ is the $D \times L$ matrix

$$\mathbf{\Gamma}^{(1)} = [\mathbf{\Gamma}_1^{(1)}, \mathbf{\Gamma}_2^{(1)}, \ldots, \mathbf{\Gamma}_L^{(1)}]. \tag{3.7}$$

The tangent vector $\dot{\mu}(0)$ to the lifted line $\mu(b)$ is a linear combination of the columns of $\mathbf{\Gamma}^{(1)}$, and so the range of the matrix $\mathbf{\Gamma}^{(1)}$ is the tangent plane $\mathbf{T}_{\mathbf{x}_0}$ of the projection manifold $\mathbf{\Omega}$ at $\mathbf{y}(\mathbf{x}_0) = \mu(0)$.

The second directional derivative (Seber and Wild, 1989) of $\mu(b)$ at $\mu(0)$ is

$$\begin{aligned}\ddot{\mu}(0) &= \left[\sum_{s=1}^{L}\frac{\partial}{\partial x_s}\left\{\sum_{r=1}^{L}\frac{\partial \mathbf{y}(\mathbf{x})}{\partial x_r}\,h_r\right\}\frac{\mathrm{d}\,x_s(b)}{\mathrm{d}\,b}\right]_{\mathbf{x}=\mathbf{x}_0,b=0} \\ &= \left[\sum_{r=1}^{L}\sum_{s=1}^{L}\frac{\partial^2\mathbf{y}(\mathbf{x})}{\partial x_r\partial x_s}\,h_r h_s\right]_{\mathbf{x}=\mathbf{x}_0} \\ &= \sum_{r=1}^{L}\sum_{s=1}^{L}\boldsymbol{\Gamma}_{r,s}^{(2)}\,h_r\,h_s, \end{aligned} \tag{3.8}$$

where $\boldsymbol{\Gamma}_{r,s}^{(2)}$ is a column vector of second-order partial derivatives of $\mathbf{y}$ (at $\mathbf{x}_0 \in \mathcal{H}$) with respect to the r-th and s-th latent space variables,

$$\begin{aligned}\boldsymbol{\Gamma}_{r,s}^{(2)} &= \mathbf{W}\Psi_{r,s}^{(2)}(\mathbf{x}_0) \\ &= \mathbf{W}\left(\frac{\partial^2\phi_1(\mathbf{x}_0)}{\partial x_r\partial x_s}, \frac{\partial^2\phi_2(\mathbf{x}_0)}{\partial x_r\partial x_s}, \ldots, \frac{\partial^2\phi_M(\mathbf{x}_0)}{\partial x_r\partial x_s}\right)^T. \end{aligned} \tag{3.9}$$

The derivatives are computed at $\mathbf{x}_0 \in \mathcal{H}$.

We decompose $\ddot{\mu}(0)$ into two orthogonal components, one lying in the tangent space $\mathbf{T}_{\mathbf{X}_0}$, the other lying in its orthogonal complement $\mathbf{T}_{\mathbf{X}_0}^{\perp}$,

$$\ddot{\mu}(0) = \ddot{\mu}^{\parallel}(0) + \ddot{\mu}^{\perp}(0), \qquad \ddot{\mu}^{\parallel}(0) \in \mathbf{T}_{\mathbf{X}_0},\ \ddot{\mu}^{\perp}(0) \in \mathbf{T}_{\mathbf{X}_0}^{\perp}. \tag{3.10}$$

The component $\ddot{\mu}^{\parallel}(0)$ describes changes in the first-order derivatives due to varying speed of parameterization, while the direction of the first-order derivatives remains unchanged. Changes in the first-order derivatives that are responsible for the curving of the projection manifold $\boldsymbol{\Omega}$ are described by the component $\ddot{\mu}^{\perp}(0)$.

$\ddot{\mu}^{\parallel}(0) = \boldsymbol{\Pi}\ddot{\mu}(0)$, where $\boldsymbol{\Pi}$ is called the orthogonal projection of $\mathcal{D}$ onto $\mathbf{T}_{\mathbf{X}_0}$. The $\boldsymbol{\Pi}$ is a linear operator described by the projection matrix

$$\boldsymbol{\Pi} = \boldsymbol{\Gamma}^{(1)}\left(\boldsymbol{\Gamma}^{(1)}\right)^{\dagger}, \tag{3.11}$$

where $\left(\boldsymbol{\Gamma}^{(1)}\right)^{\dagger}$ is the pseudo-inverse of $\boldsymbol{\Gamma}^{(1)}$. We need the orthogonal projection of $\mathcal{D}$ onto $\mathbf{T}_{\mathbf{X}_0}^{\perp}$, which is represented by $\mathbf{I} - \boldsymbol{\Pi}$, where $\mathbf{I}$ is the $D \times D$ identity matrix. So $\ddot{\mu}^{\perp}(0)$ is given by

$$\begin{aligned}\ddot{\mu}^{\perp}(0) &= (\mathbf{I} - \boldsymbol{\Pi})\,\ddot{\mu}(0) \\ &= \left[\mathbf{I} - \boldsymbol{\Gamma}^{(1)}\left(\boldsymbol{\Gamma}^{(1)}\right)^{\dagger}\right]\left[\sum_{r=1}^{L}\sum_{s=1}^{L}\boldsymbol{\Gamma}_{r,s}^{(2)}\,h_r\,h_s\right]. \end{aligned} \tag{3.12}$$

The directional curvature at $\mu(0)$ associated with the latent space direction $\mathbf{h}$ is the (Euclidean) norm of the vector $\ddot{\mu}^{\perp}(0)$. It measures the degree of local curvature of the visualisation manifold $\boldsymbol{\Omega}$ in the data space $\mathcal{D}$ (Bates and Watts, 1980). It is the *embedding curvature* of $\boldsymbol{\Omega} \subset \mathcal{D}$ at $\mathbf{y}(\mathbf{x}_0)$, evaluated with respect to the latent space direction $\mathbf{h}$.

The meaning of curvature values

By using the equations discussed in this section, it can be shown that when a two-dimension sphere

is embeded in a high-dimensional space, this definition of curvature gives the reciprocal of the sphere radius. So with a large radius, we obtain a small curvature; while a small radius will give a large curvature. Intuitively, embedding curvature may be considered as measuring the degree to which the projection manifold is curved in the data space (Amari, 1985).

Computational considerations

Since $L < D$, and usually $L = 2$ for visualisation, the computational effort needed to evaluate $\ddot{\mu}^{\perp}(0)$ must be dominated by the projection matrix $\mathbf{\Pi}$, whose scaling is $\mathcal{O}(LD^2)$. While a single evaluation of the second directional derivative $\ddot{\mu}(0)$ (for one given direction in latent space) would require $\mathcal{O}(L^2)$ operations, which is typically much less than the $\mathcal{O}(LD^2)$ scaling of $\mathbf{\Pi}$.

In practice, we consider a finite set of directions N_h so as to detect in which direction the projection manifold is maximally curved. For a total of N_h directions, the number of computational steps required to evaluate is N_h times larger than for a single direction.

3.2.2 Showing the local directional curvatures and magnification factors

- Displaying the local directional curvatures

  First, the number N_h of different latent space directions $\mathbf{h}$ with respect to which the curvatures will be computed is determined. In the case of a two-dimensional latent space, the directions $\mathbf{h}_j$, correspond to the N_h equidistant points on the unit circle, subject to the constraint that the first direction is $(1, 0)$. For the GTM model, we calculate the Euclidean norm of the directional curvature $\ddot{\mu}^{\perp}(0)$ from equation (3.12) at each latent space centre $\mathbf{x}_k$, in each direction. In the visualisation plot, we show, for each latent space centre $\mathbf{x}_k$, direction $\mathbf{h}$ yielding the maximal norm $\ddot{\mu}^{\perp}(0)$. The length of the direction line and the degree of shading of the corresponding patch are proportional to the maximal norm of $\ddot{\mu}^{\perp}(0)$.

- Showing the local magnification factors

  We evaluate the local magnification factor at each latent space centre $\mathbf{x}_k$, $k = 1, 2, ..., K$, which is the Jacobian of the GTM map $\mathbf{y}(\mathbf{x}_k; \mathbf{W})$ at $\mathbf{x}_k$ (see section 2.6.2). The magnification factor is represented by the degree of shading of the corresponding patch. In order to view plots conveniently, we use a $\log_2$ scale. So values more than 0 indicate the projection manifold is stretched in the data space, whereas values less than 0 indicate compression.

3.2.3 Experimental results

In the experiments reported here, the latent (visualisation) space of the GTM was the square $[-1, 1] \times [-1, 1]$ and the latent space centres $\mathbf{x}_k \in \mathcal{H}$ were positioned on a regular 15×15 square grid and there were 16 basis functions ϕ_m centred on a regular 4×4 square grid. Magnification factors and directional curvatures were evaluated at each latent space centre $\mathbf{x}_k$.

- Synthetic data

In the first experiment we randomly generated 1000 data points in $\mathcal{R}^3$ lying on the two-dimensional manifold shown in Figure 3.2 (a), and is defined by

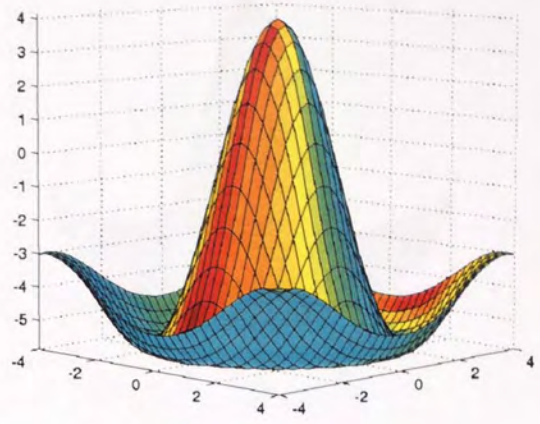
$$t_3 = \sin(r)/r, \quad \text{and } r = \sqrt{\frac{1}{2}(t_1^2 + t_2^2)}, \quad (t_1, t_2) \in [-4, 4]. \tag{3.13}$$

As expected, after training, the GTM projection manifold displayed in Figure 3.2 (b) closely followed the two-dimensional distribution of the data points. Latent space layouts of local magnification factors and directional curvatures are shown in Figures 3.2 (c) and (d), respectively. The curvature and expansion patterns in the projection manifold (Figure 3.2 (b)) are clearly reflected in the curvature and magnification factor plots. The form of the projection manifold can be approximately guessed on the basis of its local first and second order characterizations. Now, let us look at plot (d). As mentioned in section 3.2.1, it suggests that the high curvature in the middle of the plot correspond to the small value of the radius of a sphere. This is proven from plot (b), where the corresponding part of projection manifold is the top. The value of colour bar in plot (d) says the degree to which the manifold is curved when it is projected into the data space. The bigger value it is, the more curvature it forms.

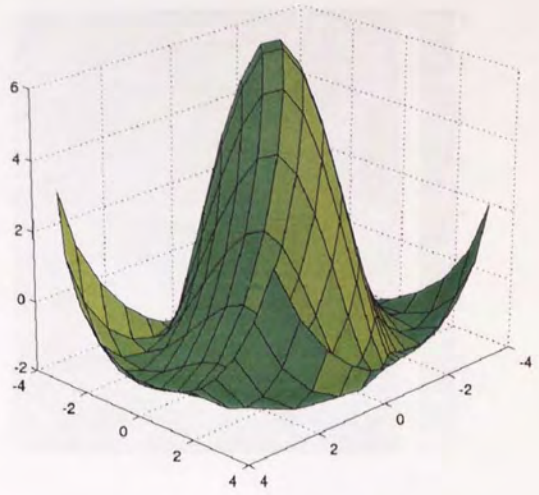
Moreover, to test the GTM's robustness, we did experiments on ten different datasets randomly generated from the same two-dimensional manifold shown in Figure 3.2 (a). The corresponding results are shown in Figure 3.3. The left column of it displays projection manifolds, the right one directional curvature plots. Looking at plots (c), (i) and (j) in Figure 3.3, we noticed the GTM model failed to capture the non-linearity of the function (3.13) three times. This is due to the dependence of EM algorithm on the initialisation. With a poor start, the EM algorithm converged to some local minima. Table 3.1 lists the corresponding average negative log-likelihood for each model. The poor models (see Figure 3.3 (c), (i) and (j)) have higher values of negative log-likelihood than the others. As for models, where the GTM successfully captured the non-linearity of the function, their curvature plots can reasonably reflect the curvatures of their corresponding projection manifolds. These results suggest that when the data distribution is known, we can use the curvature for detecting the failure of the GTM model, such as plots (c), (i) and (j). Furthermore, one may use the magnification factor and curvature plots to detect the failure of the GTM model even when the data distribution is unknown. In these cases, one can train GTM models with different parameters and then observe the magnification factors and/or curvatures to extract the similarities, which may be considered the intrinsic structures captured by the fitted models. For those models with significantly different magnification factors and/or curvatures, it may suggest that they fail to fit the data distributions.

- Oil flow data (see section 1.5.2)

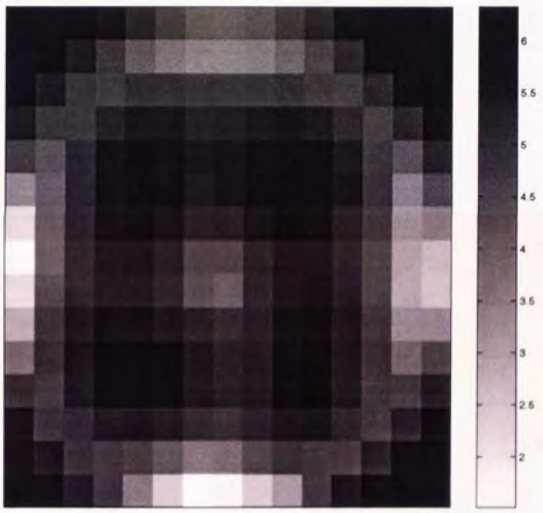
  The visualisation plot for this set is shown in Figure 3.4. Three classes were separated well. The corresponding directional curvature plot and magnification factor plot are presented in Figures 3.5 and 3.6. The curvature plot reveals that the two-dimensional projection manifold is folded



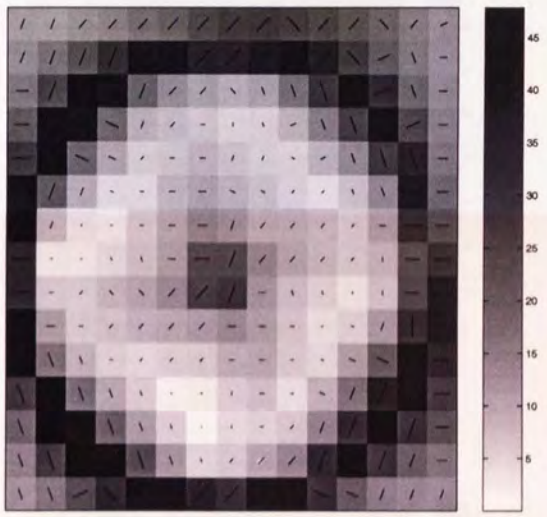
(a) Data manifold



(b) Projection manifold



(c) Magnification factors

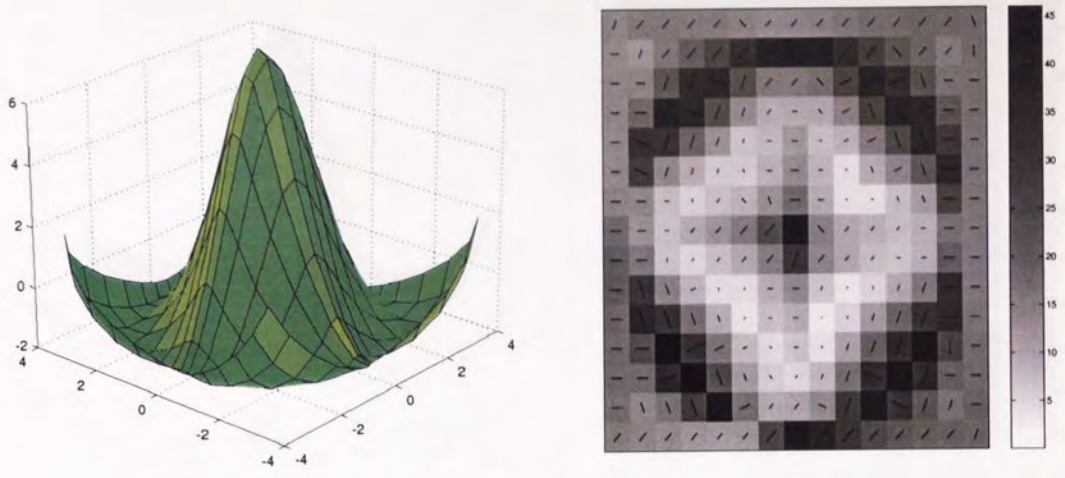


(d) Curvatures

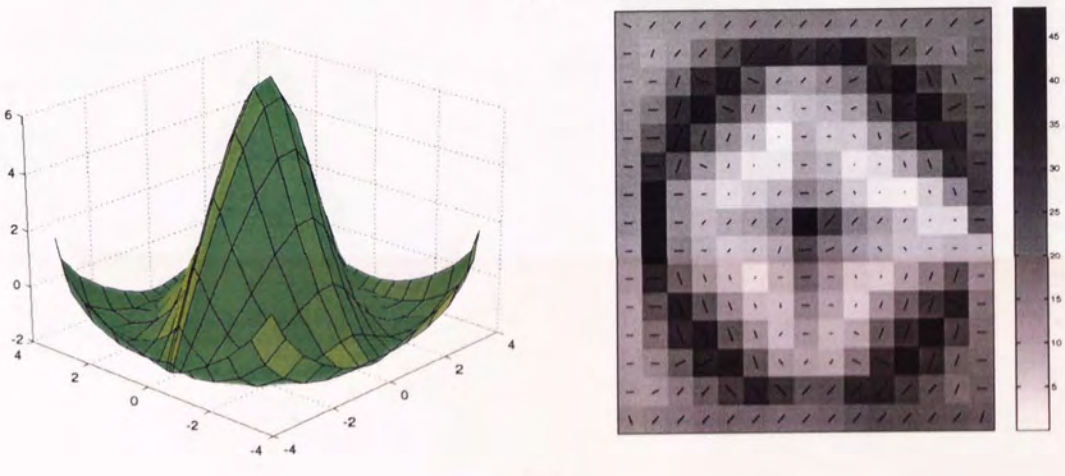
Figure 3.2: Synthetic data experiment.

	(a)	(b)	(c)	(d)	(e)
NLL	4.8216	4.8310	5.0326	4.7411	4.9019
	(f)	(g)	(h)	(i)	(j)
NLL	4.7946	5.0188	4.8977	5.2975	5.1088

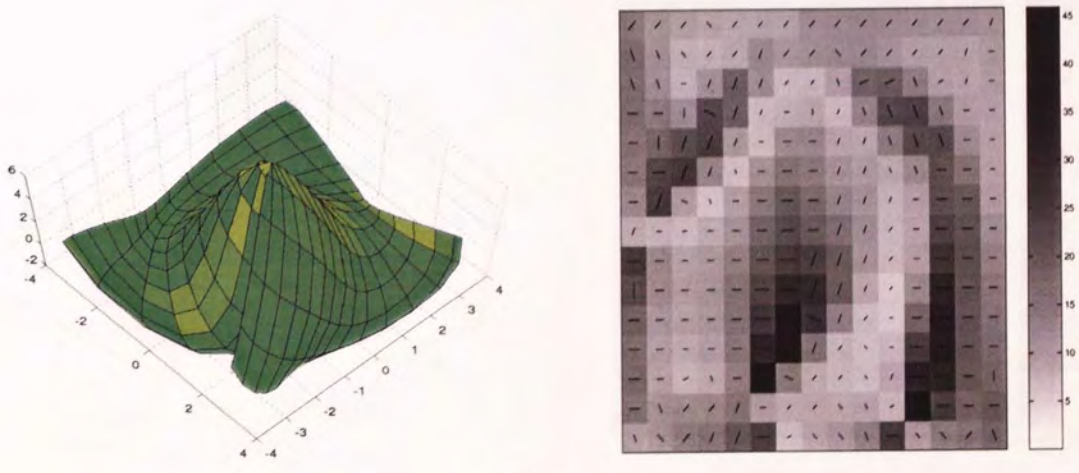
Table 3.1: Negative log-likelihood per data point



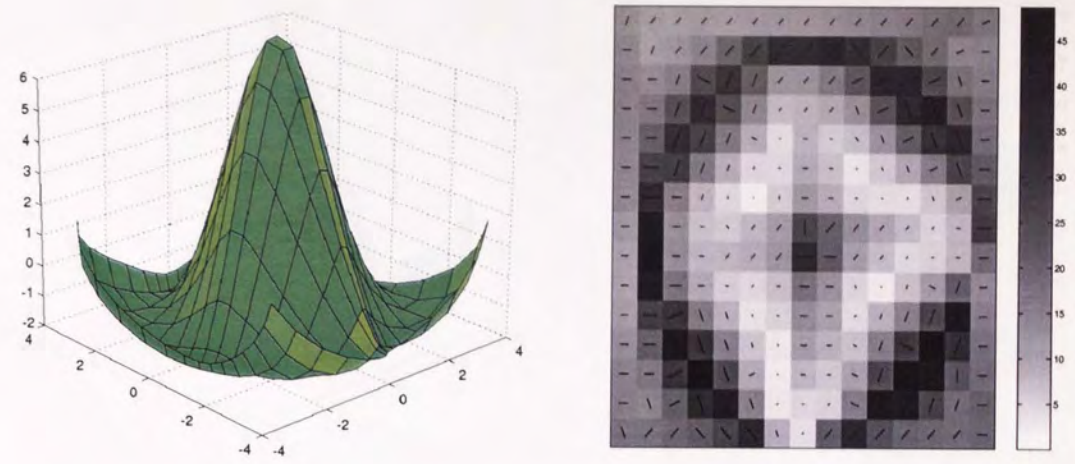
(a)



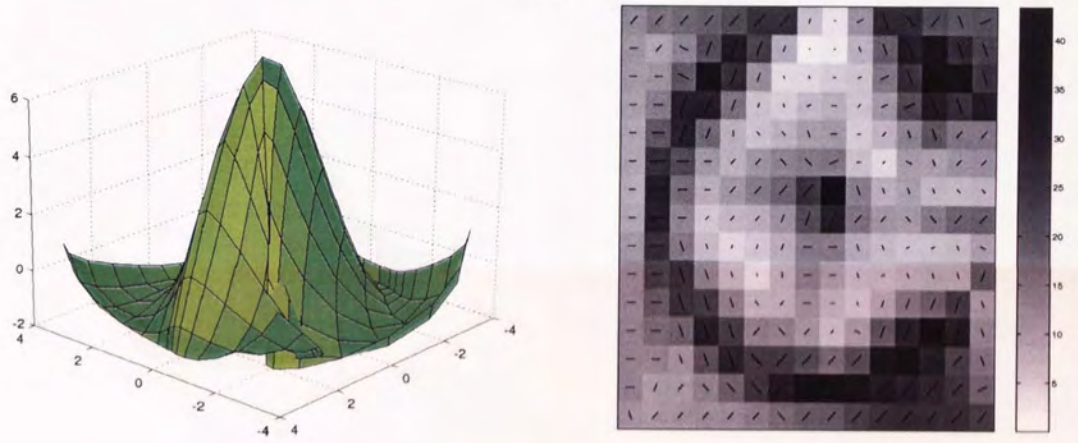
(b)



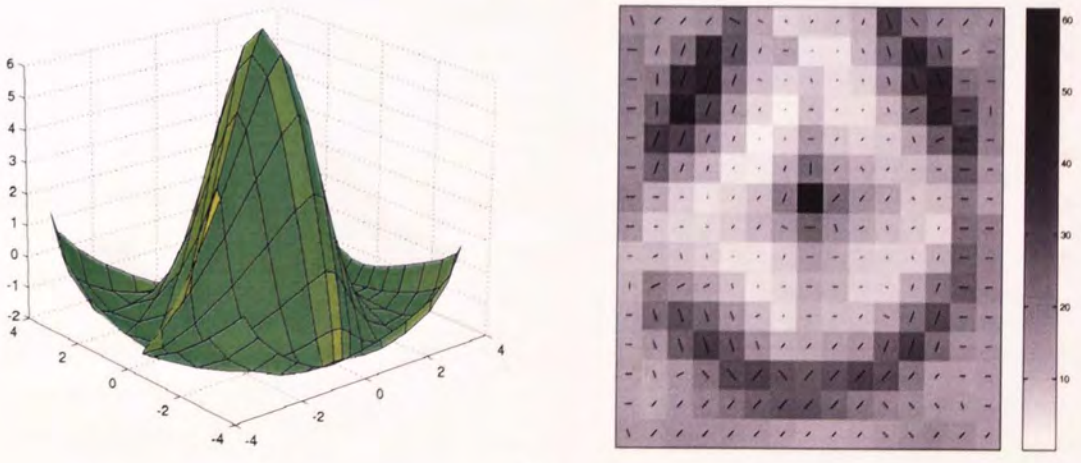
(c)



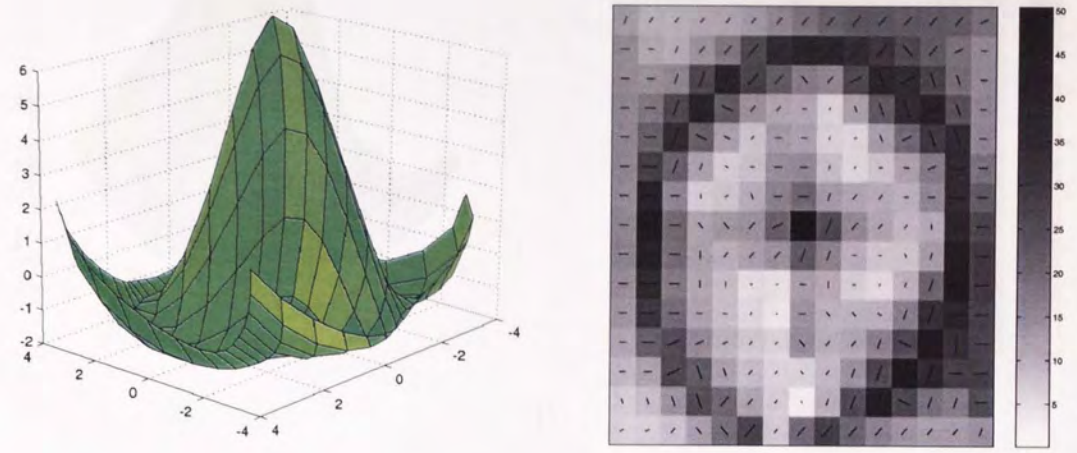
(d)



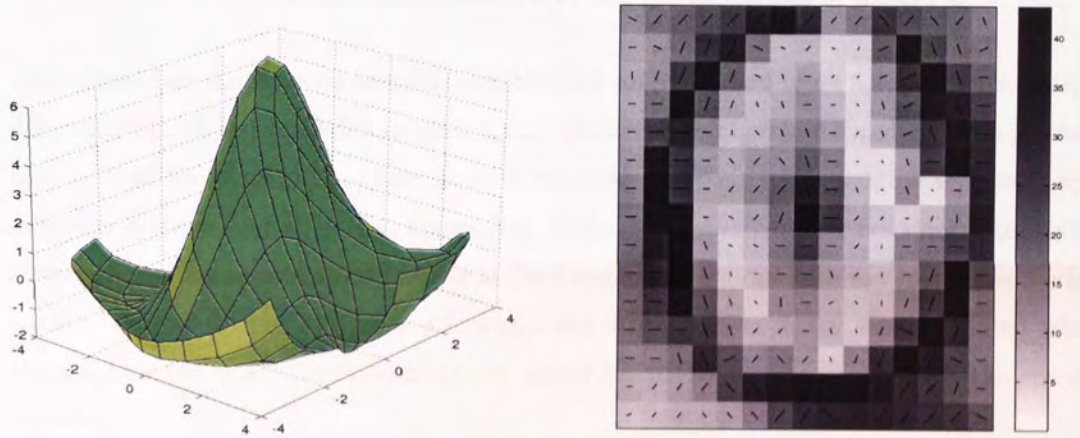
(e)



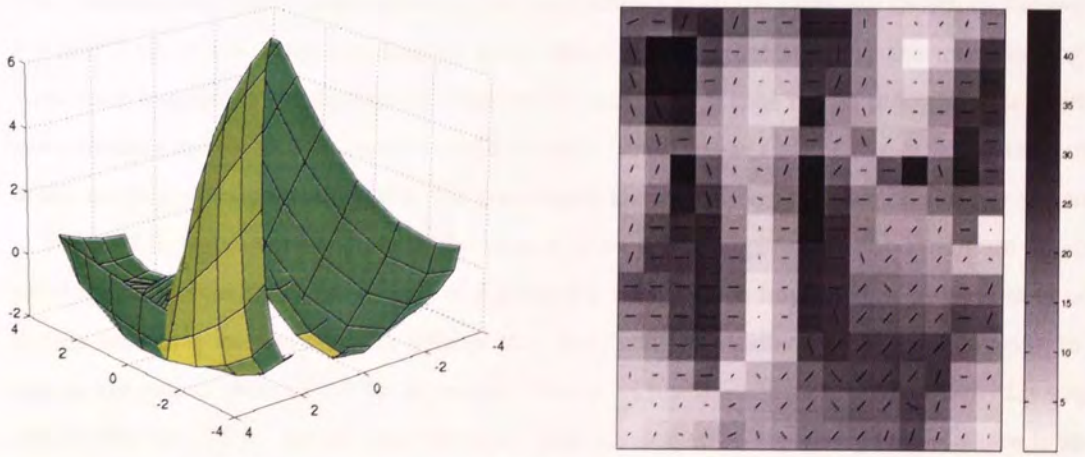
(f)



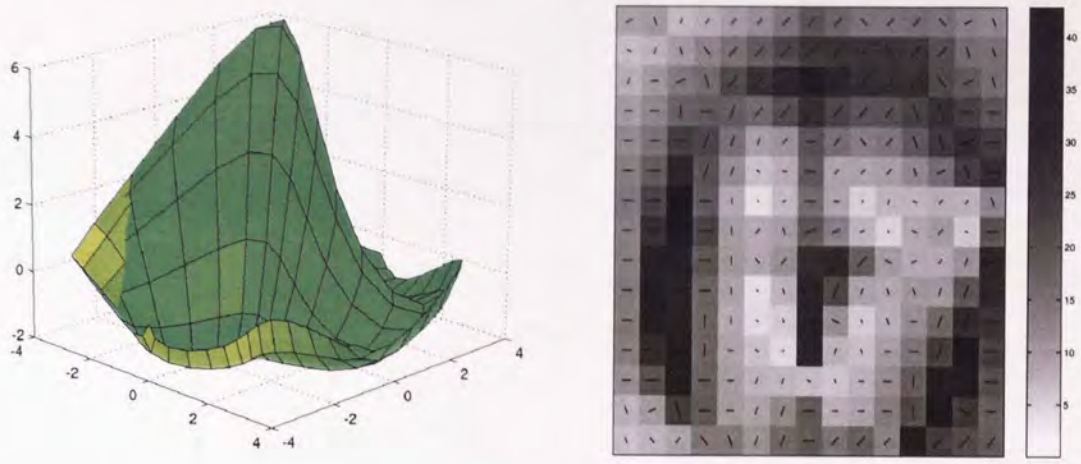
(g)



(h)



(i)



(j)

Figure 3.3: Synthetic data experiments: test the robustness of the GTM.

three times (see the pictorial annotation with 3 red solid lines in Figure 3.5) so that it can fit the data distribution in the 12-dimensional space. Referring to Figures 3.4 and 3.5, it is interesting to note that the three multi-phase flow configurations appear to be roughly separated by the folds (see three red dotted lines in Figure 3.4). It also suggests that there are approximately three clusters in the data space when looking at the compressed regions with low magnification factor values. This is obvious in Figure 3.7, where the magnification factor was integrated into the visualisation plots. In this experiment the use of curvature information is crucial to successful modeling.

- Image segmentation data (see section 1.5.3)

  The visualisation, directional curvature and magnification factor plots are shown in Figures 3.8, 3.9 and 3.10, respectively. Looking at these figures, we can see that (1) the main folding and stretching happen in the bottom of Figures 3.9 and 3.10. These might indicate that there are some outliers in this area, related to points mainly from *Grass+Foliage* appearing in the bottom of the projection visualisation plot. (2) One highly curved folding appears on the top-left corner of Figure 3.9, which corresponds to the cluster from *sky* in Figure 3.8. (3) There are no strong curving and stretching in the middle of Figures 3.9 and 3.10. It implies that properties of points in this area may be similar. We believe that the data from different classes appearing in this region are rather difficult to be separated clearly. This is verified in Figure 3.8. Looking at the middle area of it, we see that the cross and circle classes are grouped and there is strong overlapping between circle and diamond sign classes.

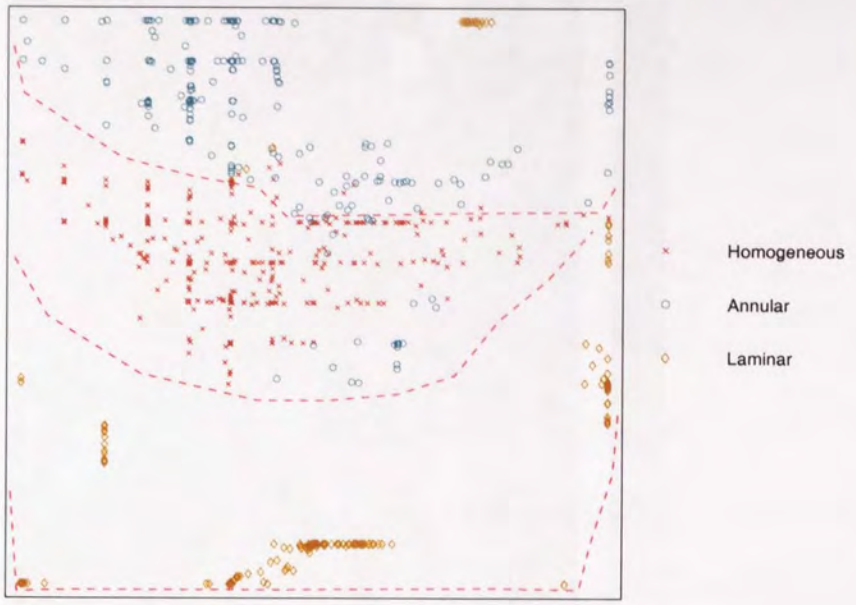


Figure 3.4: Visualisation of the GTM fitted on the oil flow data. The red dotted lines denote approximate edges of the corresponding clusters in the visualisation space.

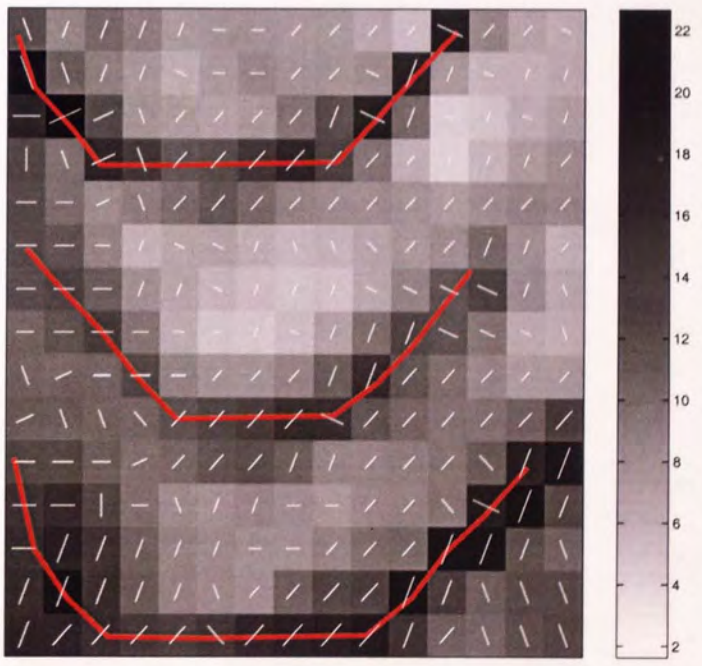


Figure 3.5: Visualisation of local curvatures of the GTM fitted on the oil flow data.The red solid lines are along the corresponding most folded regions on the projection manifold.

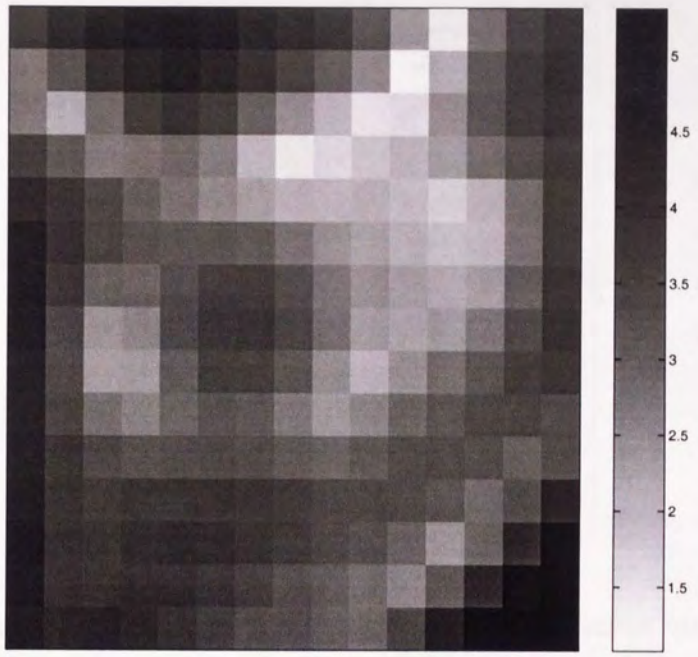


Figure 3.6: Visualisation of magnification factors of the GTM fitted on the oil flow data.

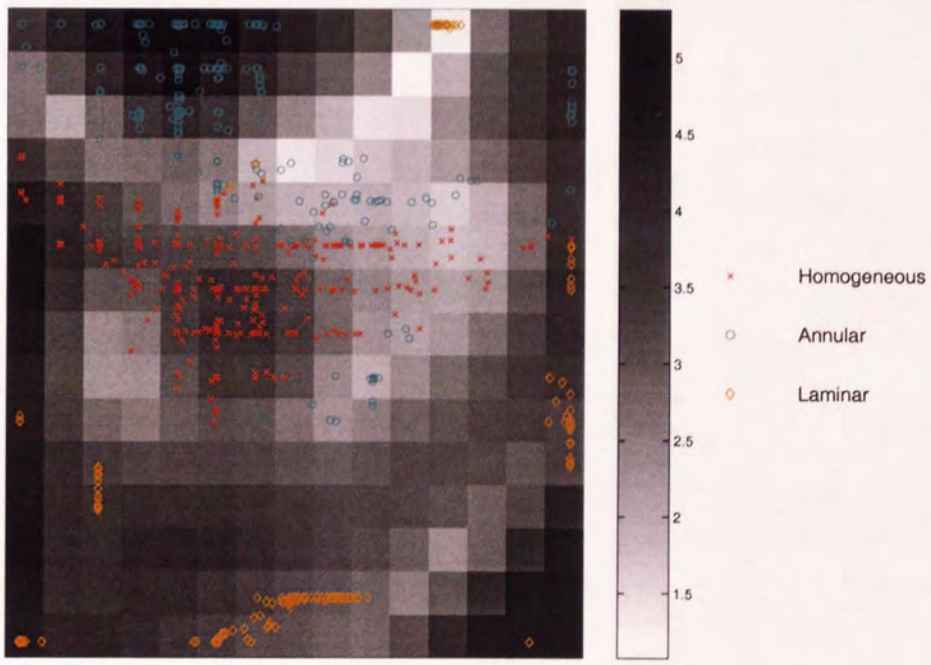


Figure 3.7: The GTM visualisation on the oil flow data with magnification factors.

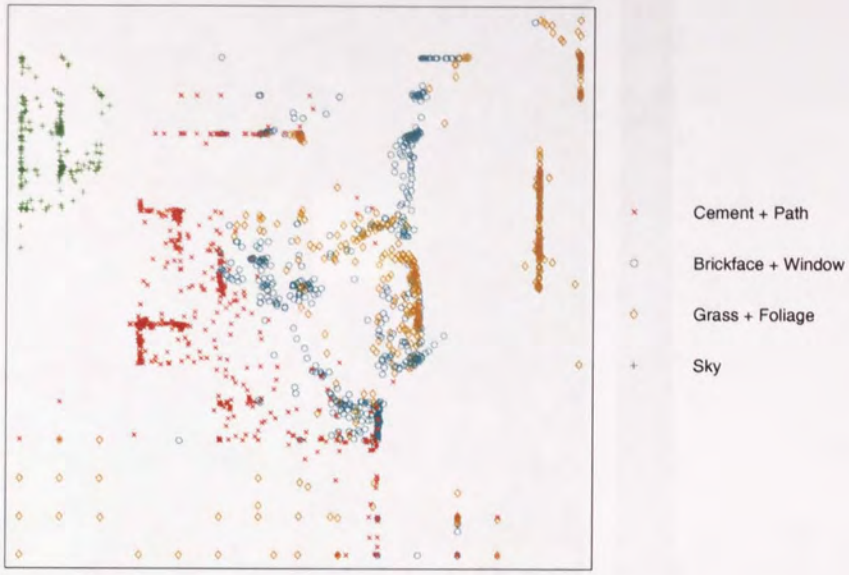


Figure 3.8: Visualisation of the GTM fitted on the image data.

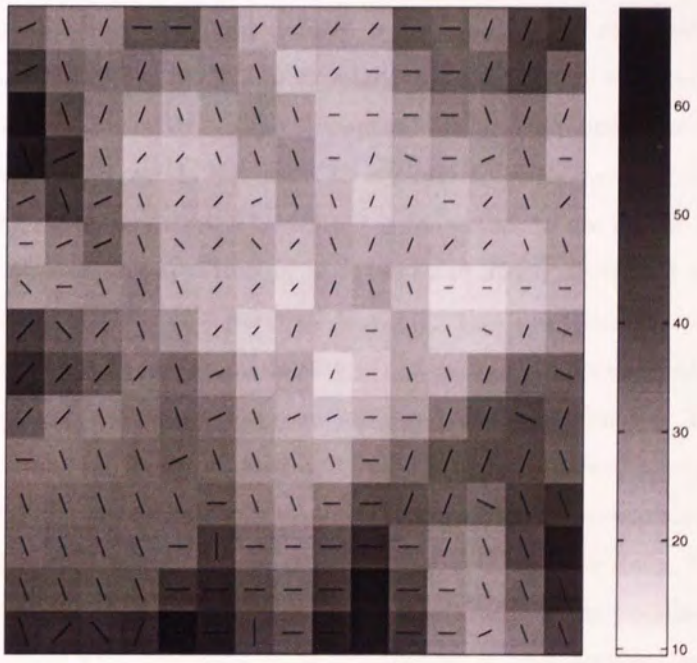


Figure 3.9: Visualisation of local curvatures of the GTM fitted on the image data.

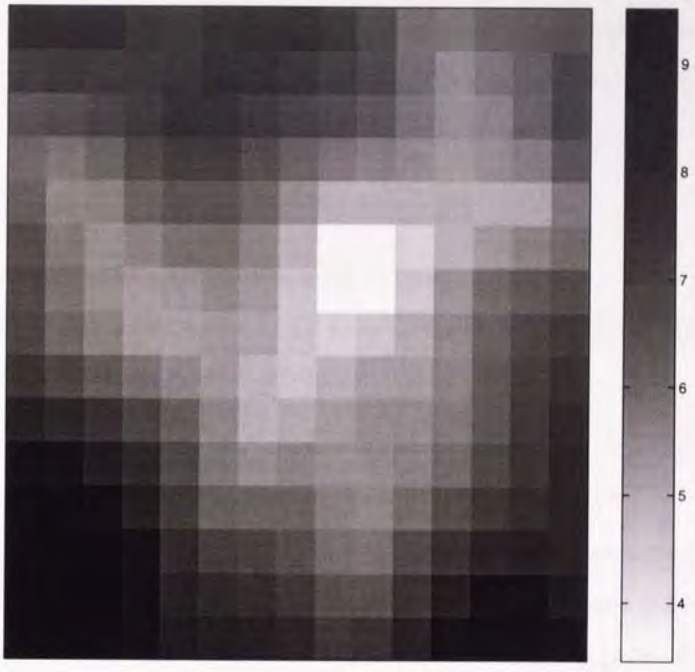


Figure 3.10: Visualisation of magnification factors of the GTM fitted on the image data.

3.3 Visualisation of incomplete data using class information constraints

In many real-world applications the input dataset is incomplete. For example, in the field of drug discovery, scientists use computer modelling to analyse the molecular structure of compounds and HTS to assess their interaction with biological targets. Many compounds are not screened against a complete set of targets, yet we do not want to exclude all such compounds from data analysis since that risks missing potential drugs. Therefore it is important to use all the available information and reconstruct the missing values. Tresp *et al.* (1994 and 1995) developed techniques for coping with missing data in the context of supervised learning: the input density $p(\mathbf{x})$ with a Gaussian mixture model and the output conditional density $p(\mathbf{t}|\mathbf{x})$ with a feedforward network are estimated, respectively. This approach maximises the joint input-output log-likelihood. The mixture model is used to integrate the likelihood over the missing inputs of the feedforward network.

The problem we consider here is how to train the GTM model with incomplete data and reconstruct the missing values using additional class information. In this way the data, including the missing components, can be shown in a visualisation plot that is as "faithful" as possible. In this section, we investigate ways of dealing with missing data in the context of unsupervised and "semi-supervised" learning.

3.3.1 Introduction

In this part, we outline the statistical framework defining missing data mechanisms which render some of data unobservable and review several basic approaches to dealing with incomplete data as well. The context presented is based on (Little, 1992).

Missing data mechanisms

There are several different mechanisms that lead to missing data, and these strongly influence the procedure used to cope with missing data. To formalise the notion of missing data mechanism, we define Ξ, a missing data indicator matrix, given by

$$\Xi_{dn} = \begin{cases} 1, & t_{dn} \text{ observed}, \\ 0, & t_{dn} \text{ missing}. \end{cases}$$

We assume that the dataset $\mathbf{T} = \{\mathbf{t}_n\}_{n=1,\ldots,N}$ can be divided into an observed component $\mathbf{T}^o$ and a missing component $\mathbf{T}^m$. Three types of missing data mechanism are distinguished by considering the conditional distribution $P(\Xi|\mathbf{T}, \varphi)$, where φ denotes a set of unknown parameters.

- Missing completely at random (MCAR).

  $P(\Xi|\mathbf{T}, \varphi) = P(\Xi|\varphi)$; that is the distribution of Ξ does not depend on the observed or missing values $\mathbf{T}$.

- Missing at random (MAR).

  $P(\Xi|\mathbf{T}, \varphi) = P(\Xi|\mathbf{T}^o, \varphi)$; that is the distribution of Ξ is related to the observed data but not to the missing data.

- Not missing at random (NMAR).

  That is $P(\Xi|\mathbf{T}^o, \mathbf{T}^m, \varphi)$ may depend on missing values.

As an example, consider a simple case. Suppose t_1, the first variable in $\mathbf{t}$, involves missing values. If the probability $P(\Xi|\mathbf{T}, \varphi)$

1. is independent of data values, then the mechanism is MCAR;

2. depends only on the values of the other variables, which are $t_2, \cdots, t_D$, then the mechanism is MAR;

3. depends on the value of t_1, then the mechanism is NMAR.

Several common techniques for analysing incomplete data

1. Complete-case analysis.

   This approach confines attention to cases where all D variables are available, which implies that cases with any missing values are simply deleted.

Advantage: Standard statistical analyses for complete-data can be used directly without modification.

Disadvantage: It is a waste of information due to discarding incomplete cases. In addition, if the MCAR assumption does not hold, bias will be introduced to parameter estimation.

2. Available-case analysis.

   This method uses the largest set of available cases in each variable to estimate each parameter. One version of available case analysis appears in (Dixon, 1983). It estimates the mean μ_d and variance σ^2_{dd} as follows:

$$\mu_d = \frac{1}{n_d} \sum_{n=1}^{n_d} t_{dn}, \tag{3.14}$$

$$\sigma^2_{dd} = \frac{1}{n_d - 1} \sum_{n=1}^{n_d} (t_{dn} - \mu_d)^2, \tag{3.15}$$

   where n_d is the number of available cases in dth variable. When estimating the covariance $\sigma^2_{dd'}$, the number $n_{dd'}$ of available cases is considered with both t_d and $t_{d'}$ observed. The covariance $\sigma^2_{dd'}$ is given by

$$\sigma^2_{dd'} = \frac{1}{n_{dd'} - 1} \sum_{n=1}^{n_{dd'}} (t_{dn} - \mu_d)(t_{d'n} - \mu_{d'}). \tag{3.16}$$

   Advantage: It can use information from incomplete cases.

   Disadvantage: The sample size is different from variable to variable. Sometime it is difficult to analyse results under this change. In addition, it also has problems of comparability across variables if the data are not MCAR.

3. Imputing unconditional means.

   This technique imputes (or fills in) missing data t_{dn} by μ_d, the unconditional sample mean of the recorded values of $\mathbf{t}_d$.

   Assuming MCAR, if the estimated covariance for observed data from available cases is $\sigma^2_{dd'}$ given by (3.16), then the variance of the observed and imputed data together is $\frac{n_d - 1}{N-1}\sigma^2_{dd}$, which is biased by a factor $\frac{n_d - 1}{N-1}$. The sample covariance of $\mathbf{t}_d$ and $\mathbf{t}_{d'}$ is biased by a factor $\frac{n_{dd'} - 1}{N-1}$.

   Figure 3.11 shows a simple example. Clearly, expectations of data with t_2 missing always lie along a horizontal line. Unconditional mean imputation ignores the covariance structure in the observed data. So it biases the estimate of the covariance.

4. Imputing conditional means.

   A more intelligent method is to fill in missing values by calculating means conditioned on observed variables, in which case the regression coefficients are functions of mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. One common method for imputing conditional means discussed in (Buck, 1960) estimates $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ based on only the complete cases, and then uses linear regression to fill in missing

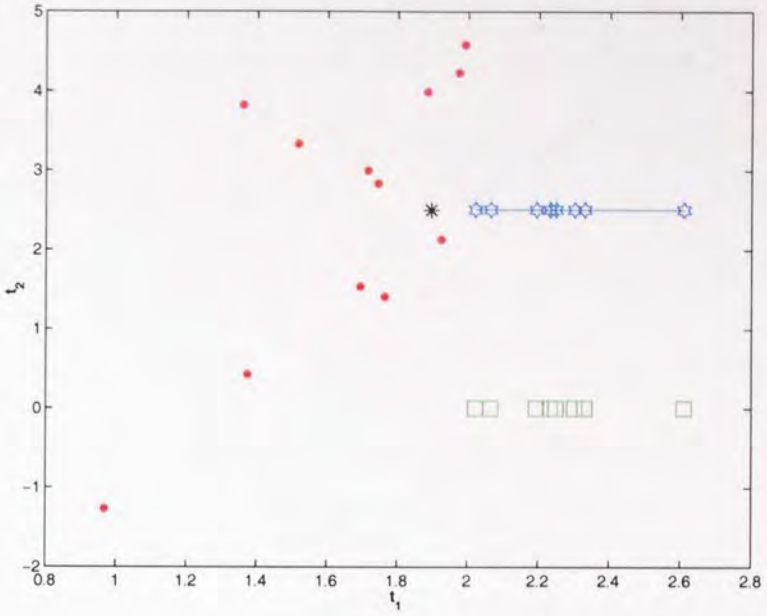


Figure 3.11: A simple example: using unconditional mean imputation to fill in missing values. Complete data were generated from a Gaussian with mean $(2, 2)$ and a diagonal covariance matrix [0.1 0; 0 2]. The points marked as dots represent cases with t_1 observed and t_2 both observed. Cases with t_1 but t_2 missing are represented by squares on the line of $t_2 = 0$. The star sign indicates the (t_1, t_2) mean calculated over the observed data. The hexagram points are obtained by unconditional mean imputation.

variables. Buck corrected for the variances but omitted the corrections for the covariances. We illustrate Buck's method for $D = 2$ variables. The result is shown in Figure 3.12.

In this case, we have

$$E[t_2|t_1] = \mu_2^{(c)} + \beta_{21}^{(c)}(t_1 - \mu_1^{(c)}), \tag{3.17}$$

where $\beta_{21}^{(c)} = \sigma_{12}^2/\sigma_{11}^2$ and (c) denotes all complete cases.

The result suggests that even this approach gives a biased estimate of the covariance matrix as the filled-in data points all fall along the regression line. On the other hand, since imputation error is not taken into account, the estimated standard errors of the regression coefficients from least square on the filled-in data will tend to be very small. Although it is possible to develop some formulas for standard errors for some special patterns, it is more difficult to derive for general patterns.

5. Maximum likelihood estimation.

   Another approach for analysing incomplete data is to compute maximum likelihood estimates for a model of the joint distribution of the complete data. Little and Rubin (1987) formulated models in terms of a joint probability distribution given by

$$P(\mathbf{T}, \Xi|\boldsymbol{\theta}, \boldsymbol{\varphi}) = P(\mathbf{T}|\boldsymbol{\theta})P(\Xi|\mathbf{T}, \boldsymbol{\varphi}), \tag{3.18}$$

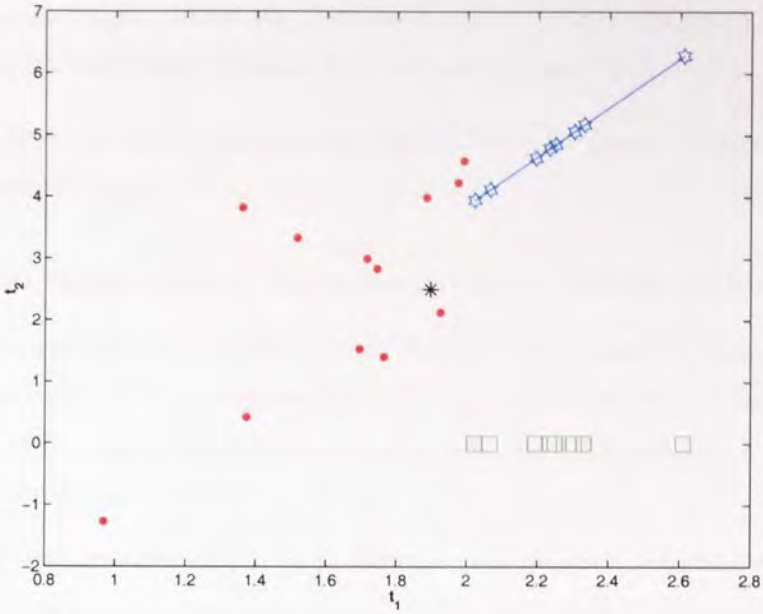


Figure 3.12: Using conditional mean imputation to fill missing values. The points marked as dots represent cases with x_1 and x_2 both observed. These points are used to calculate the least squares regression line of t_2 on t_1, that is $E[t_2|t_1] = \mu_2^{(c)} + \beta_{21}^{(c)}(t_1 - \mu_2^{(c)})$, where the superscript c signifies complete cases. Cases with t_1 observed but t_2 missing are shown by squares on the line of $t_2 = 0$. The missing variables substituted by the values calculated by Buck's method were plotted by hexagram points. The star indicates the mean calculated over the observed data.

where $\boldsymbol{\theta}$ denotes the vector of parameters for data generation process and $\boldsymbol{\varphi}$ for the missing data mechanism.

For maximum likelihood methods, the likelihood $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi})$ given the observed data is defined to be any function of $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$ proportional to $P(\mathbf{T}^o, \Xi|\boldsymbol{\theta}, \boldsymbol{\varphi})$, which is given by

$$P(\mathbf{T}^o, \Xi|\boldsymbol{\theta}, \boldsymbol{\varphi}) = \int P(\mathbf{T}^o, \mathbf{T}^m|\boldsymbol{\theta}) P(\Xi|\mathbf{T}^o, \mathbf{T}^m, \boldsymbol{\varphi})\, d\mathbf{T}^m. \tag{3.19}$$

If the missing data are MAR, that is

$$P(\Xi|\mathbf{T}^o, \mathbf{T}^m, \boldsymbol{\varphi}) = P(\Xi|\mathbf{T}^o, \boldsymbol{\varphi}), \tag{3.20}$$

then

$$\begin{aligned} P(\mathbf{T}^o, \Xi|\boldsymbol{\theta}, \boldsymbol{\varphi}) &= P(\Xi|\mathbf{T}^o, \boldsymbol{\varphi}) \int P(\mathbf{T}^o, \mathbf{T}^m|\boldsymbol{\theta})\, dT^m \\ &= P(\Xi|\mathbf{T}^o, \boldsymbol{\varphi}) P(\mathbf{T}^o|\boldsymbol{\theta}). \end{aligned} \tag{3.21}$$

Equation (3.21) states that if the data is MAR, then the likelihood can be factored. For maximum likelihood methods, it means that maximising $\mathcal{L}(\boldsymbol{\theta}|\mathbf{T}^o)$ is equivalent to maximising $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}|\mathbf{T}^o, \Xi)$.

Being a parametric method, in which a specific functional form for the density distribution is assumed, maximum likelihood estimation is not flexible as the particular form might be incapable of providing a good representation of the true density. Mixture models (see section 2.2), however,

largely overcome this problem as they combine much of the flexibility of nonparametric methods with the analytic advantages of parametric methods (Bishop, 1995).

In the next section, we shall show how the EM algorithm for Gaussian mixture models can be extended to incomplete datasets.

3.3.2 The EM algorithm for Gaussian mixture models with missing inputs

This framework was proposed in (Ghahramani and Jordan, 1994). Consider a mixture of K Gaussian models (see section 2.2.3). Binary assignment variables z_{kn} are introduced in the usual way to specify which component of the mixture generated the data point. $z_{kn} = 1$ if and only if $\mathbf{t}_n$ is generated by component k, otherwise $z_{kn} = 0$.

We write each data point $\mathbf{t}_n$ as $\begin{pmatrix} \mathbf{t}_n^o \\ \mathbf{t}_n^m \end{pmatrix}$, where m and o denote sub-vectors and sub-matrices of the parameters matching the missing and observed components of the data and each data vector can have a different pattern of missing components, which means the missing values maybe appear in different positions for each data point.

To handle missing data, the EM algorithm combining both the assignment variables and the missing inputs $\mathbf{T}^m$ can be written as follows:

- $E-$step: to compute the expectation of the error function $\langle E_{comp} \rangle$, where E_{comp} given by equation (2.6) can be rewritten as:

$$E_{comp} = -\sum_{n=1}^{N}\sum_{k=1}^{K} z_{kn} \log P(k) - \sum_{n=1}^{N}\sum_{k=1}^{K} z_{kn} \log p(\mathbf{t}_n|\boldsymbol{\theta}_k). \tag{3.22}$$

  We can ignore the first term since we only estimate the parameters of $p(\mathbf{t}_n|\boldsymbol{\theta}_k)$. For the Gaussian distribution with the full covariance matrix (see equation (2.18)), we can expand the second term of the right hand side in equation (3.22):

$$\begin{aligned} E_{comp} \quad = \quad & \sum_{n=1}^{N}\sum_{k=1}^{K} z_{kn}[\frac{1}{2}\log|\boldsymbol{\Sigma}_k| + \frac{D}{2}\log 2\pi \\ & + \frac{1}{2}(\mathbf{t}_n^o - \boldsymbol{\mu}_k^o)^T \boldsymbol{\Sigma}_k^{-1,oo}(\mathbf{t}_n^o - \boldsymbol{\mu}_k^o) \\ & + (\mathbf{t}_n^o - \boldsymbol{\mu}_k^o)^T \boldsymbol{\Sigma}_k^{-1,om}(\mathbf{t}_n^m - \boldsymbol{\mu}_k^m) \\ & + \frac{1}{2}(\mathbf{t}_n^m - \boldsymbol{\mu}_k^m)^T \boldsymbol{\Sigma}_k^{-1,mm}(\mathbf{t}_n^m - \boldsymbol{\mu}_k^m)] \end{aligned} \tag{3.23}$$

  where $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ denote the means and covariances of the kth Gaussian respectively. We use m and o superscripts to denote subvectors and submatrices of the parameters matching the missing and observed components of the data. Note that the superscript, for example, $(-1, oo)$ denotes inverse followed by submatrix operations; $\boldsymbol{\Sigma}_k$ is divided into $\begin{pmatrix} \boldsymbol{\Sigma}_k^{oo} & \boldsymbol{\Sigma}_k^{om} \\ \boldsymbol{\Sigma}_k^{mo} & \boldsymbol{\Sigma}_k^{mm} \end{pmatrix}$ corresponding to $\mathbf{t} = \begin{pmatrix} \mathbf{t}^o \\ \mathbf{t}^m \end{pmatrix}$

- $M-$step: to update parameters by minimising the expected error $\langle E_{comp} \rangle$ with respect to the parameters

$$\boldsymbol{\theta}^j = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \langle E_{comp} \rangle. \tag{3.24}$$

The expected value in the $E-$step is taken with respect to both sets of missing variables. After taking the expectation, the sufficient statistics for the parameters include three unknown terms, z_{kn}, $z_{kn}\mathbf{t}_n^m$ and $z_{kn}\mathbf{t}_n^m\mathbf{t}_n^{m^T}$, so we must calculate the expectations for these three terms. To compute these expectations, variables $\hat{t}_{kn}^m$ are introduced,

$$\hat{\mathbf{t}}_{kn}^m \equiv \langle \mathbf{t}_n^m | z_{kn} = 1, \mathbf{t}_n^o, \boldsymbol{\theta}_k \rangle = \boldsymbol{\mu}_k^m + \boldsymbol{\Sigma}_k^{mo}\boldsymbol{\Sigma}_k^{oo^{-1}}(\mathbf{t}_n^o - \boldsymbol{\mu}_k^o), \tag{3.25}$$

which is the least-squares linear regression between $\mathbf{t}_n^m$ and $\mathbf{t}_n^o$ predicted by the kth Gaussian. Expectation of z_{kn} is $\langle z_{kn} | \mathbf{t}_n^o, \boldsymbol{\theta}_k \rangle = R_{kn}$, which is defined as follows, measured only on the observed dimensions of $\mathbf{t}_n$.

$$R_{kn} = \frac{|\boldsymbol{\Sigma}_k|^{-1/2} \exp\left\{-\frac{1}{2}(\mathbf{t}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{t}_n - \boldsymbol{\mu}_k)\right\}}{\sum_{k'=1}^{K} |\boldsymbol{\Sigma}_{k'}|^{-1/2} \exp\left\{-\frac{1}{2}(\mathbf{t}_n - \boldsymbol{\mu}_{k'})^T \boldsymbol{\Sigma}_{k'}^{-1}(\mathbf{t}_n - \boldsymbol{\mu}_{k'})\right\}} \tag{3.26}$$

As for the second and third terms, we get

$$\langle z_{kn}\mathbf{t}_n^m | \mathbf{t}_n^o, \boldsymbol{\theta}_k \rangle = \langle z_{kn} | \mathbf{t}_n^o, \boldsymbol{\theta}_k \rangle \langle \mathbf{t}_n^m | z_{kn} = 1, \mathbf{t}_n^o, \boldsymbol{\theta}_k \rangle = R_{kn}\hat{\mathbf{t}}_{kn}^m, \tag{3.27}$$

and

$$\begin{aligned} \langle z_{kn}\mathbf{t}_n^m\mathbf{t}_n^{m^T} | \mathbf{t}_n^o, \boldsymbol{\theta}_k \rangle &= \langle z_{kn} | \mathbf{t}_n^o, \boldsymbol{\theta}_k \rangle \langle \mathbf{t}_n^m\mathbf{t}_n^{m^T} | z_{kn} = 1, \mathbf{t}_n^o, \boldsymbol{\theta}_k \rangle \\ &= R_{kn}(\boldsymbol{\Sigma}_k^{mm} - \boldsymbol{\Sigma}_k^{mo}\boldsymbol{\Sigma}_k^{oo^{-1}}\boldsymbol{\Sigma}_k^{mo^T} + \hat{\mathbf{t}}_{kn}^m\hat{\mathbf{t}}_{kn}^{m^T}). \end{aligned} \tag{3.28}$$

After computing these expectations in the $E-$step, the $M-$step uses them substituted into equations (2.20) and (2.23) to reestimate the means and covariances. We substitute the values of $\hat{\mathbf{t}}_{kn}^m$ for the missing values of $\mathbf{t}_n$ to reestimate the mean vector in equation (2.20). We substitute the values of $(\boldsymbol{\Sigma}_k^{mm} - \boldsymbol{\Sigma}_k^{mo}\boldsymbol{\Sigma}_k^{oo^{-1}}\boldsymbol{\Sigma}_k^{mo^T} + \hat{\mathbf{t}}_{kn}^m\hat{\mathbf{t}}_{kn}^{m^T})$ for the outer product matrices involving the missing data of $\mathbf{t}_n$ to reestimate the covariance matrix in equation (2.23). Note that there are simplifications for Gaussian distribution with non-full covariance matrix. In this case, variables $\hat{\mathbf{t}}_{kn}^m$ are simplified to be equal to $\boldsymbol{\mu}_k^m$ in equation (3.25).

3.3.3 Incorporating missing values into the EM algorithm for the GTM model

We can extend the Gaussian mixture model EM algorithm to the GTM as follow. In the $E-$step for the GTM, the expectation of z_{kn} is $\langle z_{kn} | \mathbf{t}_n^o, \boldsymbol{\theta}_k \rangle = R_{kn}$, where

$$R_{kn} = \frac{\left(\frac{\beta}{2\pi}\right)^{D/2} \exp\left\{-\frac{\beta}{2}||\mathbf{y}(\mathbf{x}_k; \mathbf{W}) - \mathbf{t}_n||^2\right\}}{\sum_{k'}^{K}\left(\frac{\beta}{2\pi}\right)^{D/2} \exp\left\{-\frac{\beta}{2}||\mathbf{y}(\mathbf{x}_{k'}; \mathbf{W}) - \mathbf{t}_n||^2\right\}}, \tag{3.29}$$

measured only on $\mathbf{t}_n^o$, the observed dimensions of $\mathbf{t}_n$.

As mentioned in section 2.5, the GTM itself is a Gaussian mixture model. The mean $\boldsymbol{\mu}_k$ of a single Gaussian k in the data space is equal to the image $\mathbf{y}(\mathbf{x}_k; \mathbf{W})$ of the latent grid node $\mathbf{x}_k$, mapped using an RBF network. Following equation (3.25), we introduce

$$\begin{aligned} \hat{\mathbf{t}}_{kn}^m &= \langle \mathbf{t}_n^m | z_{kn} = 1, \mathbf{t}_n^o, \boldsymbol{\theta}_k \rangle \\ &= (\mathbf{y}_k^m)^{old} + \boldsymbol{\Sigma}_k^{mo} \boldsymbol{\Sigma}_k^{oo^{-1}} (\mathbf{t}_n^o - (\mathbf{y}_k^o)^{old}), \end{aligned} \tag{3.30}$$

which is the least-squares regression between $\mathbf{t}_n^m$ and $\mathbf{t}_n^o$ predicted by the kth Gaussian, and 'old' denotes the value computed in the last $M-$step, $(\mathbf{y}_k^m)^{old} = (\mathbf{W}_{old}\boldsymbol{\Phi}(\mathbf{x}_k))^m$ and $(\mathbf{y}_k^o)^{old} = (\mathbf{W}_{old}\boldsymbol{\Phi}(\mathbf{x}_k))^o$. As the covariance matrix of the GTM is isotropic, $\Sigma_k = \beta^{-1}\mathrm{I}$, and the covariance of missing and observed values Σ_k^{mo} is equal to 0. So we have:

$$\hat{\mathbf{t}}_{kn}^m = (\mathbf{y}_k^m)^{old}. \tag{3.31}$$

In the $M-$step, the weights are updated to $\mathbf{W}_{new}$ as described in equation (2.59) for complete training data, which has the form as follows:

$$\mathbf{W}_{new}\boldsymbol{\Phi}\mathbf{G}\boldsymbol{\Phi}^T = \mathbf{T}\boldsymbol{R}^T\boldsymbol{\Phi}^T, \tag{3.32}$$

where the missing values are filled in with the posterior means,

$$\langle \mathbf{t}_n^m | \mathbf{t}_n^o, \boldsymbol{\theta}_j \rangle = \sum_{k=1}^{K} R_{kn} \hat{\mathbf{t}}_{kn}^m. \tag{3.33}$$

The inverse variance is updated using the following formula:

$$\beta^{-1} = \frac{1}{\mathrm{ND}} \sum_{n=1}^{N} \sum_{k=1}^{K} R_{kn} \left(\|\mathbf{t}_n^o - \mathbf{y}_k^o\|^2 + \langle z_{kn} \|\mathbf{t}_n^m - \mathbf{y}_k^m\|^2 \rangle \right), \tag{3.34}$$

where

$$\begin{aligned} \langle z_{kn} \|\mathbf{t}_n^m - \mathbf{y}_j^m\|^2 \rangle &= n_m (\beta^{-1})^{old} + (\hat{\mathbf{t}}_{kn}^m)^T (\hat{\mathbf{t}}_{kn}^m) - 2(\hat{\mathbf{t}}_{kn}^m)^T \mathbf{y}_k^m \\ &\quad + (\mathbf{y}_k^m)^T \mathbf{y}_k^m, \end{aligned} \tag{3.35}$$

and n_m is the number of missing values in data point $\mathbf{t}_n$; $\mathbf{y}_k^o$ and $\mathbf{y}_k^m$ are computed using $(\mathbf{W}_{new}\boldsymbol{\Phi}(\mathbf{x}_k))^o$ and $(\mathbf{W}_{new}\boldsymbol{\Phi}(\mathbf{x}_k))^m$. A more detailed derivation can be found in Appendix A.

3.3.4 Learning with class-conditional information

When visualising a set of data points $T = \{\mathbf{t}_1, ..., \mathbf{t}_n\}$ from a set of class-labelled points $T_c = \{(\mathbf{t}_1, c_1), (\mathbf{t}_2, c_2), ..., (\mathbf{t}_N, c_N)\}$, with class labels c_n from a set $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_J\}$, one can use the class information as a clue for reasoning about missing values in the corrupted data points $\mathbf{t}_n$. Given a corrupted point $\mathbf{t}_n$, instead of computing the responsibilities $R_{kn} = P(\mathbf{x}_k | \mathbf{t}_n^o)$, we determine the *class-conditional responsibilities*

$$\begin{aligned} R_{knc} &= P(\mathbf{x}_k | \mathbf{t}_n^o, c_n) \\ &= \frac{P(\mathbf{x}_k, c_n | \mathbf{t}_n^o)}{\sum_{k'=1}^{K} P(\mathbf{x}_{k'}, c_n | \mathbf{t}_n^o)}. \end{aligned} \tag{3.36}$$

By the (standard) assumption of conditional independence of observed variables, given the hidden ones, we have

$$p(\mathbf{t}_n^o, c_n|\mathbf{x}_k) = p(\mathbf{t}_n^o|\mathbf{x}_k)P(c_n|\mathbf{x}_k). \tag{3.37}$$

Using a flat prior on the latent space centres $P(\mathbf{x}_k) = 1/K$, $k = 1, 2, ..., K$,

$$\begin{aligned} P(\mathbf{x}_k, c_n|\mathbf{t}_n^o) &= \frac{p(\mathbf{t}_n^o, c_n|\mathbf{x}_k)P(\mathbf{x}_k)}{\sum_{k'=1}^{K}\sum_{C_i\in\mathcal{C}} p(\mathbf{t}_n^o, C_i|\mathbf{x}_{k'})P(\mathbf{x}_{k'})} \\ &= \frac{p(\mathbf{t}_n^o|\mathbf{x}_k)P(c_n|\mathbf{x}_k)P(\mathbf{x}_k)}{\sum_{k'=1}^{K}\sum_{C_i\in\mathcal{C}} p(\mathbf{t}_n^o|\mathbf{x}_{k'})P(C_i|\mathbf{x}_{k'})P(\mathbf{x}_{k'})} \\ &= R_{kn}\ P(c_n|\mathbf{x}_k). \end{aligned} \tag{3.38}$$

The distribution of class labels, conditioned on the latent space centres $\mathbf{x}_k$, is computed by determining the "mass" of uncorrupted training points "explained" by $\mathbf{x}_k$ and belonging to a class $C_i \in \mathcal{C}$,

$$P(C_i|\mathbf{x}_k) = \frac{\sum_{\mathbf{t}_n\in T_{comp}; c_n=C_i} p(\mathbf{t}_n|\mathbf{x}_k)}{\sum_{\mathbf{t}_n\in T_{comp}} p(\mathbf{t}_n|\mathbf{x}_k)}, \tag{3.39}$$

where T_{comp} is a collection of uncorrupted training points. This means we are assuming that the mechanism for missing data does not depend on the class.

Using equations (3.37) and (3.38), (3.36) can be rewritten as

$$\begin{aligned} R_{knc} &= \frac{p(\mathbf{t}_n^o|\mathbf{x}_k)\ P(c_n|\mathbf{x}_k)}{\sum_{k'=1}^{K} p(\mathbf{t}_n^o|\mathbf{x}_{k'})\ P(c_n|\mathbf{x}_{k'})} \\ &= \frac{p(\mathbf{t}_n^o|\mathbf{x}_k)\ P(\mathbf{x}_k|c_n)}{\sum_{k'=1}^{K} p(\mathbf{t}_n^o|\mathbf{x}_{k'})\ P(\mathbf{x}_{k'}|c_n)}, \end{aligned} \tag{3.40}$$

where for $C_i \in \mathcal{C}$,

$$P(\mathbf{x}_j|C_i) = \frac{P(C_i|\mathbf{x}_k)}{\sum_{k'=1}^{K} P(C_i|\mathbf{x}_{k'})}. \tag{3.41}$$

It follows from (3.40), that unlike the original latent centres' responsibilities R_{kn}, where a flat prior $P(\mathbf{x}_k) = 1/K$ is imposed, the class-conditional responsibilities, R_{knc}, are calculated using a *class-conditional prior on latent space*, $P(\mathbf{x}_k|C_i)$, $C_i \in \mathcal{C}$. When faced with corrupted data points, such class-conditional priors help us to eliminate (at least to some degree) multi-modalities in the posterior over the latent space centres. This may be no longer a true EM algorithm. However, in our experiments there was no occasion on which the likelihood decreased during training, even with large numbers of missing values.

3.3.5 Summary of the algorithm

Briefly, our algorithm can be described as follows: a density model (GTM) of the data is learned in an unsupervised way from the incomplete training data using an EM algorithm where sufficient statistics for the missing data are estimated from equations (3.29) and (3.35) in the $E-$step. Note that the responsibility R_{kn} in equation (3.29) is measured only on the observed dimensions of $\mathbf{t}_n$. For visualisation purposes, the missing values are filled in by computing the means of their conditional distributions given by equation (3.33). In the $M-$step, the weights are updated using equation

(3.32); the inverse variance is updated using equation (3.34). This generic algorithm can be improved by taking into account the class label information associated with each point in the training set. In this case, we can obtain the probability from equation (3.39) that a particular class c was responsible for generating the latent space centre. By using Bayes' theorem, we can compute the class-conditional priors from equation (3.41) for all latent space centres, which can then be used to calculate the posterior probabilities (or 'responsibilities') of the latent space centres for the incomplete data points, given by equation (3.40).

3.3.6 Experimental results

In this section we compare three training algorithms for the GTM with missing data: **(I)** EM for missing data using class information as described in Section 3.3.4; **(II)** standard EM on the data first completed by class-conditional mean imputation. In this case, each missing component of a data point $\mathbf{t}_n$ from class C_i is filled by simply calculating the average of all observed values of the same component (dimension) in the points from the class C_i; **(III)** EM for missing data (i.e. with on-line estimation of missing values but not using class information, as described in section 3.3.3).

Synthetic data

We first consider a synthetic dataset, which consists of 80 training points lying on the curve $t_2 = \sin(t_1)$. The coordinate t_1 was generated uniformly in the interval $[-\pi/2, 7\pi/2]$, and spherical Gaussian noise with standard deviation 0.1 was added to the t_2 coordinate. We have defined 4 classes as shown in Figure 3.13. We also sampled a (complete) test dataset of 200 data points from the same distribution.

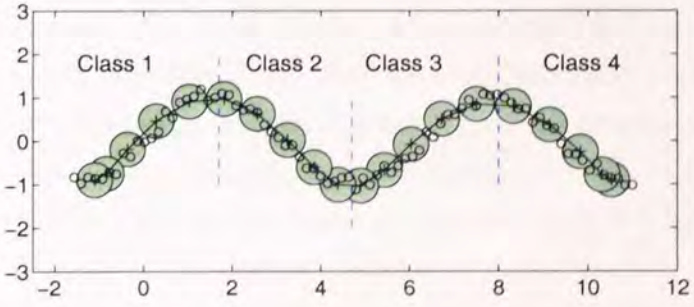


Figure 3.13: The GTM at iteration 20 trained on complete dataset

Some values of t_1 were removed at random from points in all classes. A straightforward application of the generic algorithm (III) gave unsatisfactory results because of multiple modes in the responsibilities (2.58). For example, if $t_2 = 0$, t_1 could be $0, \pi, 2\pi$ or 3π with equal probability. The network was also trained using Algorithms I and II. The proportion of missing values was varied from 0 to 70 percent. For each percentage of missing values, the experiments were carried out 10 times deleting different values at random.

Plots (a) and (b) in Figure 3.14 display the results using Algorithms I and II. The estimated

missing values and original values for each data point are shown in plots and they are joined by a red line. If an algorithm works well, the estimated missing values and their corresponding original values will be close together and the red lines between them will be very short. It suggests that the EM algorithm combining the class-conditional information is highly successful, while the missing values estimated using class-conditional mean imputation lie on straight lines. Note how the greater uncertainty in estimating missing data for Algorithm II leads to much larger variance in the trained GTM. Plot (c) displays the test set likelihood. It shows that the network can perform better using Algorithm I than using Algorithm II and still performs well with even up to 60% missing values in the training dataset.

Oil Flow Data

In this experiment, we randomly chose 200 data points from each class in the oil pipeline flow dataset for training, while the remainder constitutes the test set. A GTM with a two-dimensional latent space was used to model and visualise the data. In each set, 50% of the data points in each class are incomplete, with between 6 and 9 of the 12 values removed.

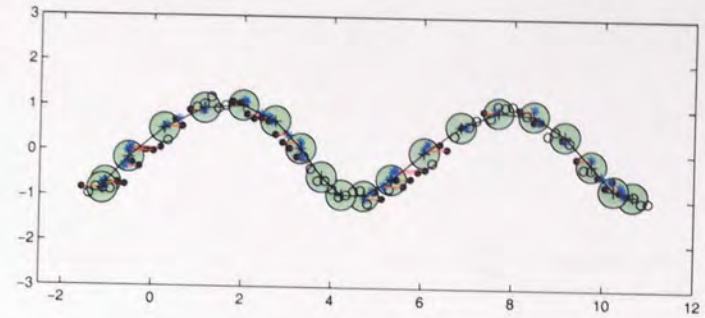
Figure 3.15 displays the results on the training set. It suggests that Algorithm I can be an improvement over the two other algorithms for missing data. Plot (b) shows better separation of classes and matches better to the result obtained from the complete dataset (plot (a)). After using class-conditional mean imputation, some strongly overlapped clusters appear in plot (c) since the same means are substituted for missing values of the same class. As for plot (d), which was obtained just by the generic algorithm, the homogeneous and annular classes are not separated well as we did not use the class-conditional prior knowledge in the training process.

Figure 3.16 presents projections of test data points with trained GTM models. Looking at this figure, the model trained by Algorithm I (plot (b)) has a good generalization performance. Again, rather than having significant overlaps (see plot (c)), plot (b) presents better separation of classes and matches better to the result obtained from the complete dataset (plot (a)). As for the result with the generic algorithm (see plot (d)), it is similar to plot (b), however, the points from class *homogeneous* and *annular* at the bottom-left corner were not separated.

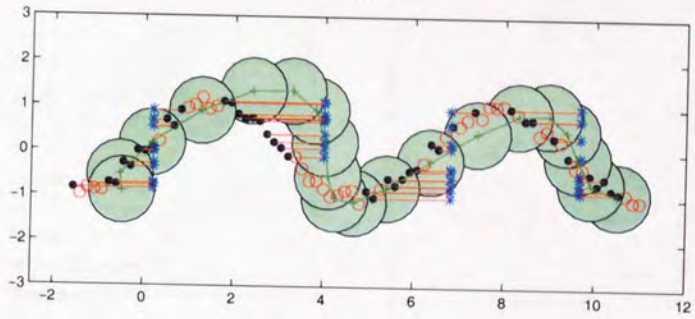
We see from Figures 3.15 and 3.16 that Algorithm I has better training results than the others, however, its average negative log-likelihood on test dataset is a bit greater than Algorithm II (see Table 3.2). Because this result was unexpected given the training set visualisation, we looked in more detail at two models trained with Algorithms I and II. We plotted each test data point probability density

	Complete dataset	Algorithm I	Algorithm II	Algorithm III
NLL	-4.2334	-1.4422	-1.5908	-0.5066

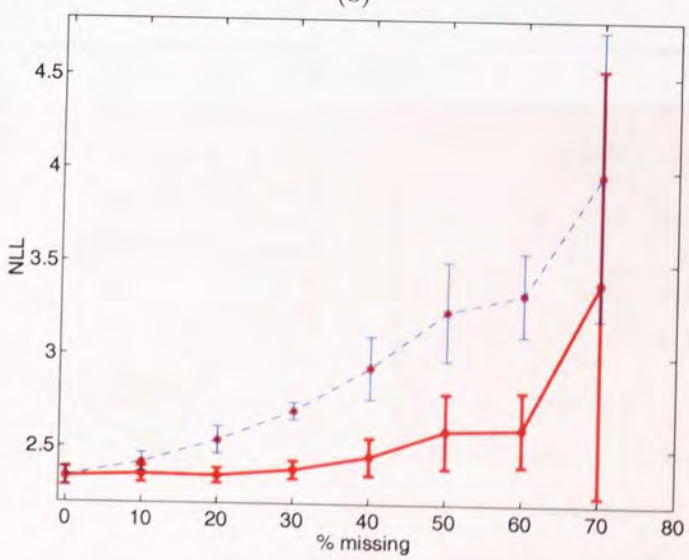
Table 3.2: Average negative log-likelihood on oil test dataset with different algorithm.



(a)



(b)



(c)

Figure 3.14: Synthetic problem: illustrations of GTMs at iteration 20 ((a)–(b)) trained on data with 50% of t_1 values missing. Complete points are plotted as circles. Stars represent the estimated missing values, while the dots show the original values. The centres of the GMM are plotted as plus signs, and are joined by a line according to their ordering in the (one-dimensional) latent space (K = 20). The discs surrounding each plus sign represent two standard deviations' width of the noise model. The estimated missing values and original values for each data point are joined by a red line. (c) The average test set negative log-likelihood (NLL) and standard deviation error bars for 10 repetitions of the training process. Algorithm I and II are represented by the thick solid and dashed lines, respectively.

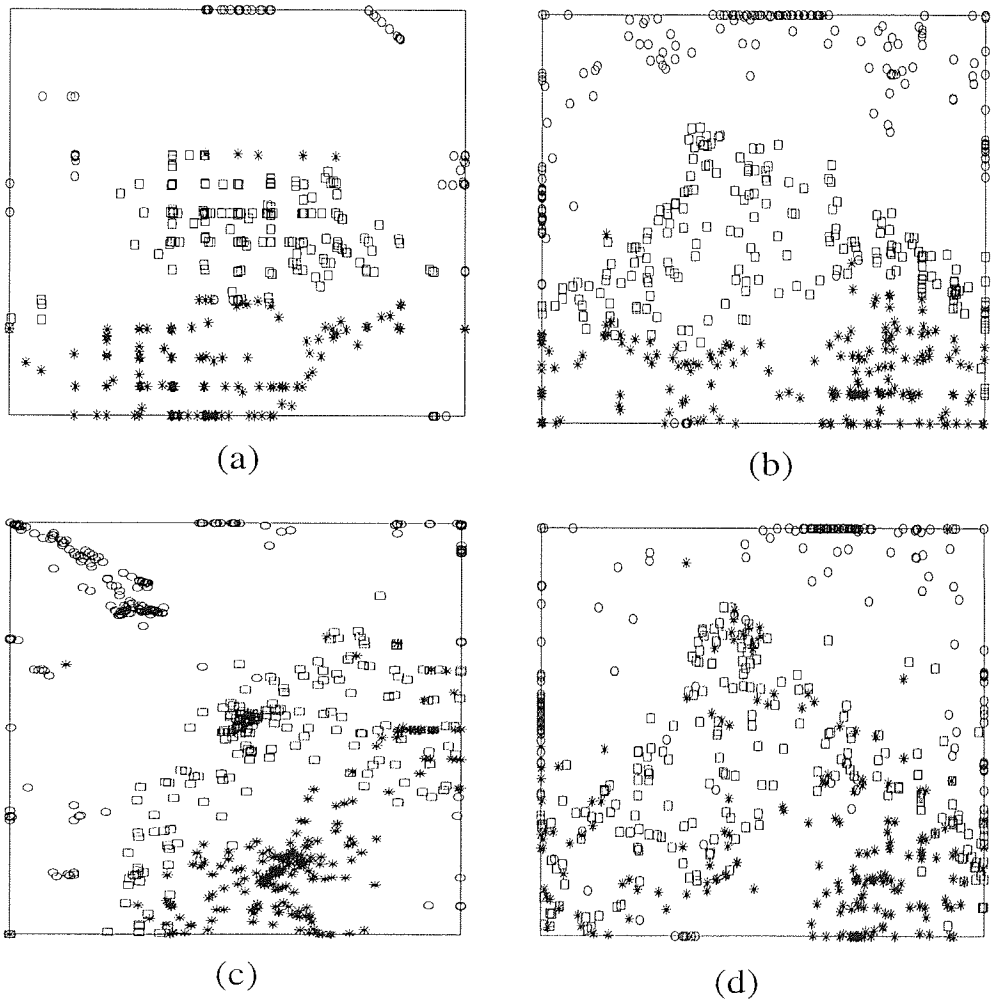


Figure 3.15: Oil flow training set: (a) training on the complete dataset. The remaining plots show results for models trained on incomplete data. (b) EM algorithm combining the class-conditional information (Algorithm I); (c) standard EM using conditional mean imputation (Algorithm II); (d) generic EM without class information (Algorithm III). The classes homogeneous, annular and laminar are represented by square, star and circle signs respectively.

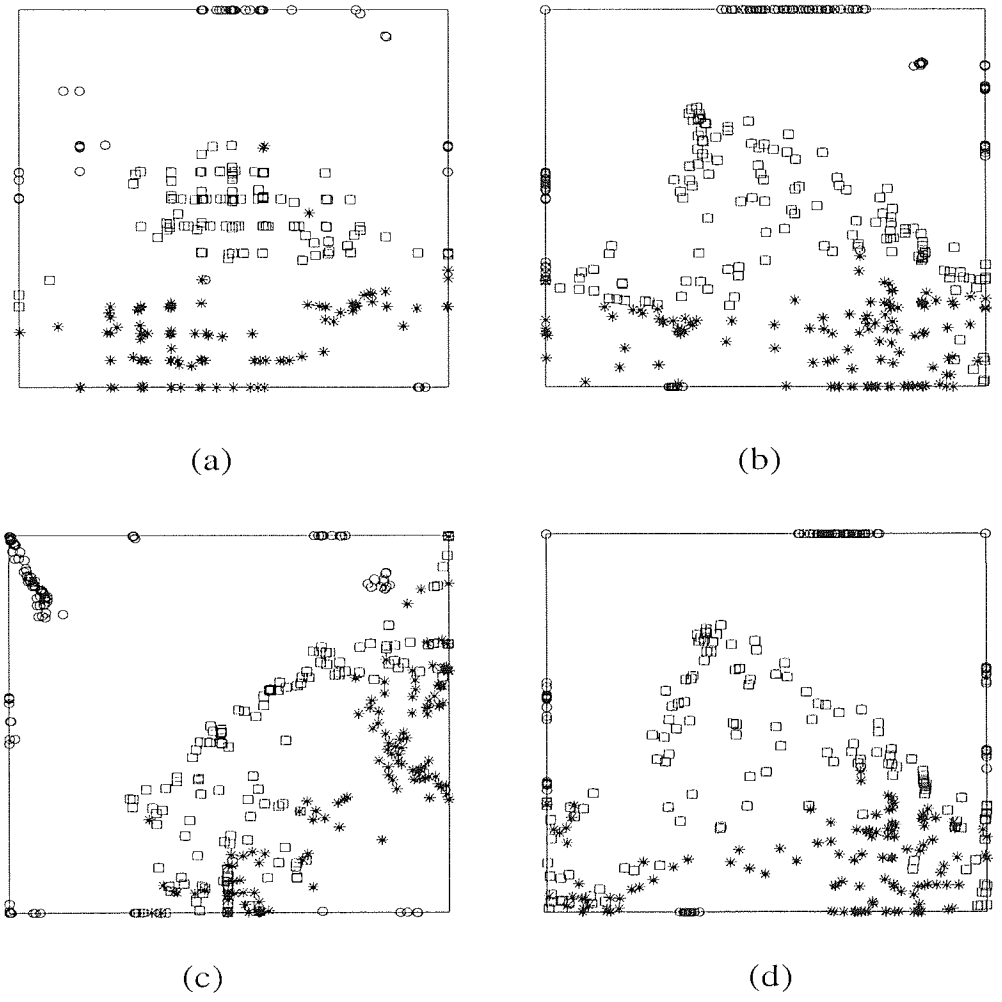
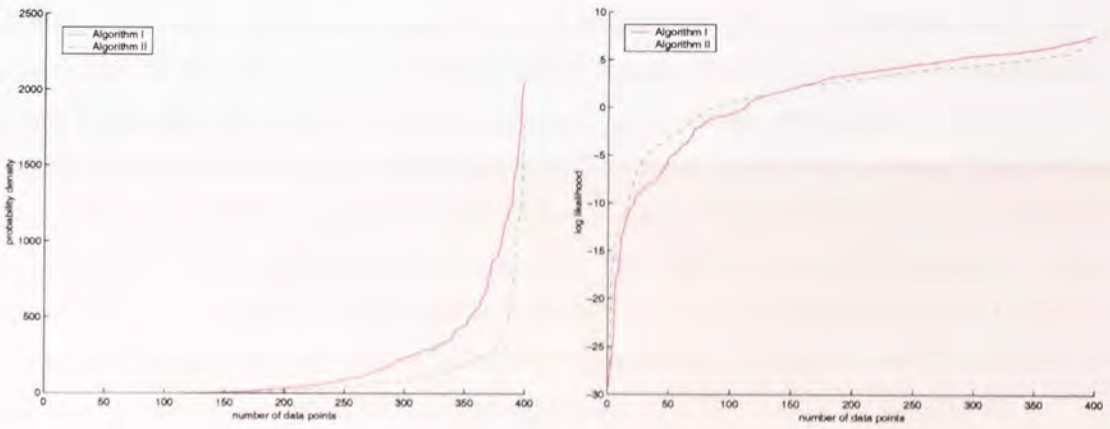


Figure 3.16: Oil flow test dataset: (a) training on the complete dataset. The remaining plots show results for models trained on incomplete data. (b) EM algorithm combining the class-conditional information (Algorithm I); (c) standard EM using conditional mean imputation (Algorithm II); (d) EM without class information (Algorithm III). The classes homogeneous, annular and laminar are represented by square, star and circle signs respectively.

and log-likelihood under these two models. The results are displayed in Figure 3.17. Differences (see plot (a)) between these two models are decreased after taking the log scale (see plot (b)). We see from plot (a) that for the first 150 data points the two models have similar performance, but probability density values of these data points are very low (close to 0). Figure 3.18 presents histograms on the two models. It shows that there are more data points with relative smaller density values in Algorithm II. Algorithm II is better than Algorithm I for these badly filled points, so the negative log-likelihood contributions are smaller. But we do not care about the badly filled points, since for example, it is not real difference between the density being 0.0005 (Algorithm II) and 0.00005 (Algorithm I).

Since the GTM is a constrained mixture of Gaussians, it does not fit data perfectly when just using a simple two dimensional manifold. Outliers (e.g. the values of log-likelihood were less than 0) have a large effect on the negative log-likelihood and these are points far from the manifold. The point is how well it fit most of points. In addition, we compared the variances of these two models. The variance of model trained using Algorithm I was 0.0210, while one from model trained by Algorithm II was 0.0244. The larger variance in Algorithm II gave lower negative log-likelihood on these points.



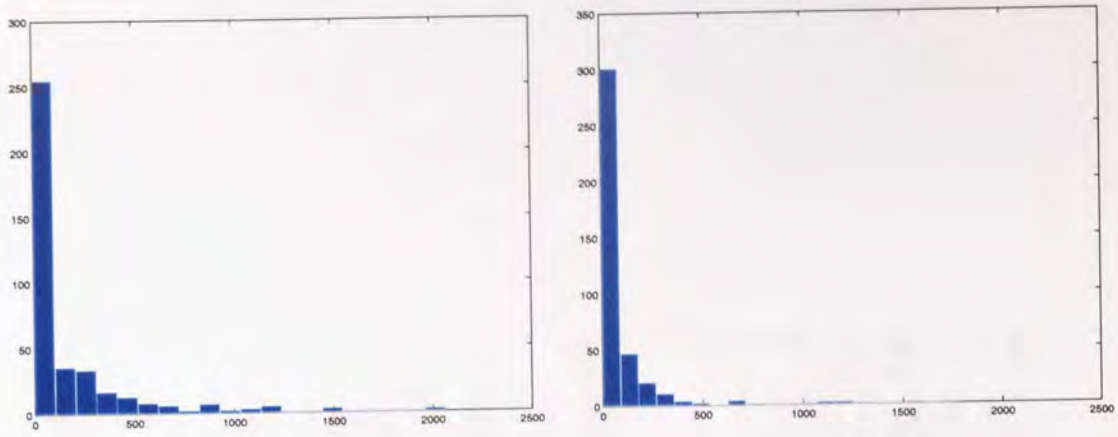
(a) (b)

Figure 3.17: (a) probability density per data point. (b) log-likelihood per data point.

Image Data

In the last experiment, our dataset is from the image segmentation dataset. We divided this set into two sub-sets, the training set including 60% of data points from each class and the test set including the remainder. Again, 50% of data points in each class are incomplete, with between 6 and 9 of 18 values removed.

Figure 3.19 shows the results on the training set. With reference to plot (a), reconstructions with Algorithm I (see plot (b)) give a better projections on *Sky* class on the top left of the plot, in comparison with Algorithm II (see plot (c)). Looking at *Cement+Path* and *Brickface+Window* classes in these plots, it is shown that data points from these classes give much more separation between data



(a) (b)

Figure 3.18: Histograms show probability density of data points: (a) fitted using the class-conditional information (Algorithm I); (b) fitted using conditional mean imputation (Algorithm II).

points on the middle part of the plot (b), By contrast, Algorithm II (plot (c)) has more significant overlaps on the data from these classes than those corresponding projections in plot (a), since it filled in missing values of the same class by using the same means. Thus it is difficult to assess which algorithm is better than the others. For the test set, the case is similar (see Figure 3.20).

To obtain more information on models trained with different algorithms, average negative log-likelihoods on the test dataset are listed in Table 3.3. As shown, errors are all very large. Moreover, we plotted probability densities against each test point for Algorithm I and II displayed in Figure 3.21. We see that even the largest density value is less than 0.1, which means that all the GTMs are equally poor with projection manifolds far away from data points. It suggests that it is difficult to just use a two dimensional manifold to fit the whole dataset in a 18 dimensional data space.

	Complete dataset	Algorithm I	Algorithm II	Algorithm III
NLL	13.3983	13.8711	13.4316	14.0877

Table 3.3: Average negative log-likelihood on image test dataset with different algorithm

3.4 Discussion

In this chapter we have extended the GTM model in two directions: **(1)** using tools from differential geometry we derived expressions for local directional curvatures of the projection manifold; **(2)** we gave the details of the derivation of the missing data algorithm; furthermore, we modified the training algorithm for the GTM using class information to improve results on incomplete data.

Curvature plots are useful for detecting regions where geometry is bent, for guiding the user in making changes in the amount of regularization in non-linear projection manifold, and for choosing

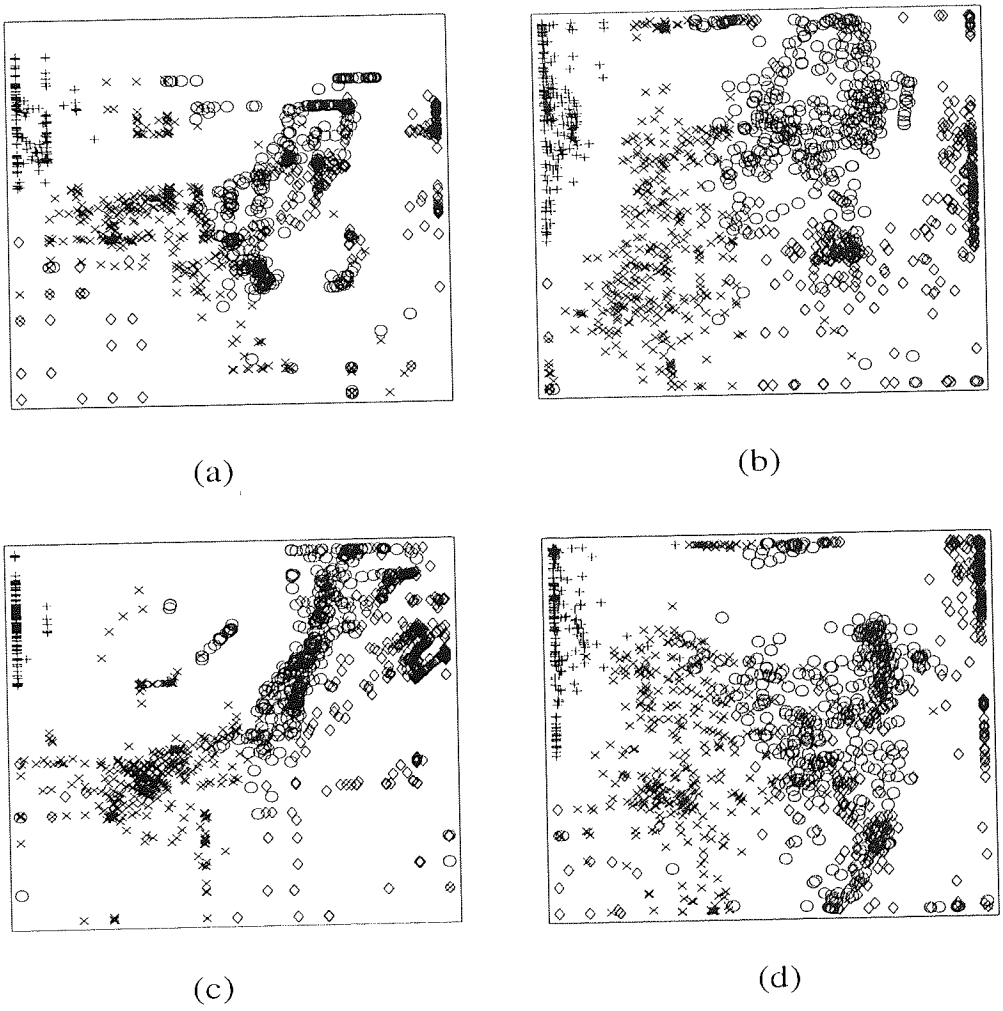


Figure 3.19: Image training dataset: (a) training on the complete dataset. The remaining plots show results for models trained on incomplete data. (b) EM algorithm combining the class-conditional information (Algorithm I); (c) standard EM using conditional mean imputation (Algorithm II); (d) generic EM without class information (Algorithm III). The classes Cement+Path, Brickface+Window, Grass+Foliage and Sky are represented by cross, circle, diamond and plus signs respectively.

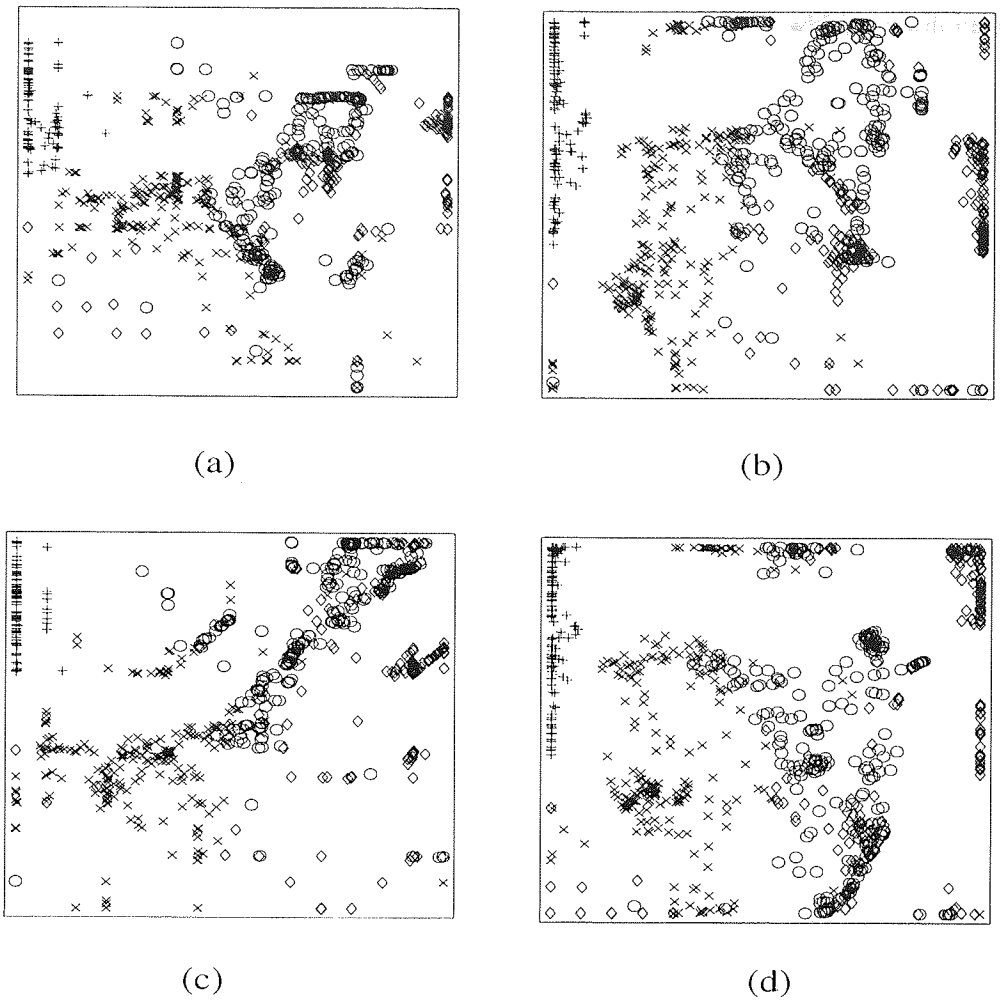


Figure 3.20: Image test set: (a) training on the complete dataset. The remaining plots show results for models trained on incomplete data. (b) EM algorithm combining the class-conditional information (Algorithm I); (c) standard EM using conditional mean imputation (Algorithm II); (d) generic EM without class information (Algorithm III). The classes Cement+Path, Brickface+Window, Grass+Foliage and Sky are represented by cross, circle, diamond and plus signs respectively.

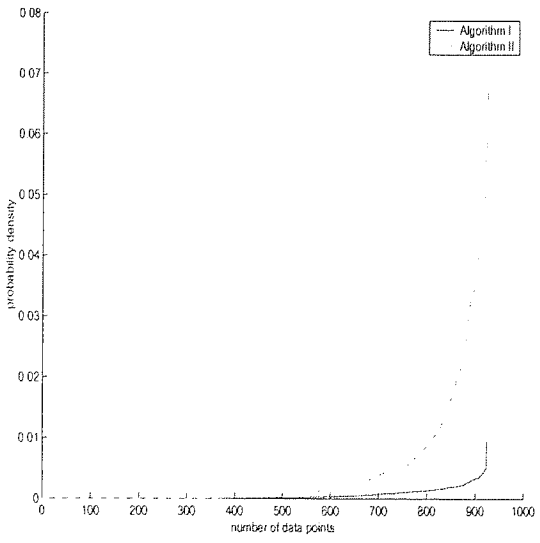


Figure 3.21: Probability density against each data point

regions of interest when constructing detailed low-level visualisation plots, which we will introduce in chapter 4.

In section 3.3, we have shown how incomplete data and class information can both be included in the GTM training process. Consider a special case, where each class is a simple (Gaussian-like) cluster in the data space. In this case, we believe that filling in the missing values by using the simple class-conditional mean imputation can obtain a good visualisation result. However, if there exist many incomplete data in each class, the plot will have significant overlaps. On the other hand, if there are several clusters, for example, 2 well-separated dense clusters in each class in the data space, then the class mean of this multimodal distribution can be a low probability point, in which case using the simple class-conditional mean imputation is inappropriate. The new algorithm is preferable to the simple strategy of just filling in the missing values with conditional means. The problem of multiple modes in the GTM responsibilities for missing data can be overcome (at least to some degree) by means of the class-conditional prior and this significantly improves both the visualisation plot and the fit of the model.

Chapter 4

Hierarchical Latent Trait Models

This chapter describes the hierarchical latent trait model (HLTM) with noise models from the exponential family of distributions. The basic building block is the latent trait model (LTM). We have developed an interactive visualisation software system based on this visualisation algorithm, which is useful for data analysis and data mining in high dimensional data space. We have applied the HLTM to three datasets and compared it with a hierarchical (linear) latent variable model. Employing hierarchical latent trait models in HTS dataset is a new application.

4.1 Introduction

It is known that a single two-dimensional visualisation plot is not sufficient to capture all of the interesting information when datasets are complex. For example, the 18 dimensional image dataset fitted by a simple two-dimension projection manifold shown in section 3.3.6. In that case, the projection manifold was far away from the data points. This has motivated researchers to develop hierarchical visualisation systems, which allow the complete dataset to be visualised at the top level, with subclusters of data points visualised at deeper levels. In Bishop and Tipping (1998), authors proposed a locally linear hierarchical visualisation system, while in Tiňo and Nabney (2002), the model was extended to non-linear projection manifolds, but with a Gaussian noise model (i.e. with the GTM as the basic component).

Since the latent trait model has a probabilistic formulation, it is straightforward to provide an extended EM algorithm to build the hierarchical tree. Kaban *et al.* (2002) briefly gave a general formulation of hierarchical latent trait mixture models. In this chapter we review their work and give more details. Furthermore, this chapter is devoted to actual applications including text mining and HTS data analysis. From the hierarchical visualisation approach, we can learn more internal structures of a dataset with refined and detailed representations at the deeper levels. Readers will see how this visualisation system helps us to learn useful knowledge from data.

In the next section, we will briefly introduce a mixture of LTMs. The generalised formulation for

hierarchical LTMs is provided in section 4.3. Section 4.4 describes the hierarchical LTM visualisation implementation. We will present some experimental results on "real-world" datasets in section 4.5, where the application of this sort of visualisation in HTS is new. This chapter will be ended in section 4.6 by summarising some key issues.

4.2 Mixtures of latent trait models

Before we set up the hierarchical model, let us consider a *mixture* of A latent trait models. The corresponding density function $p(\mathbf{t})$ is:

$$p(\mathbf{t}) = \sum_{a=1}^{A} P(a)p(\mathbf{t}|a), \tag{4.1}$$

where $P(a)$ are the non-negative mixture coefficients (prior probabilities), corresponding to the mixture components $p(\mathbf{t}|a)$ and satisfying $\sum_{a=1}^{A} P(a) = 1$, and $p(\mathbf{t}|a)$ is a distribution on a data space $\mathcal{D}$, $\mathbf{t} \in \mathcal{R}^D$, defined by each model a,

$$p(\mathbf{t}|a) = \sum_{k=1}^{K_a} p(\mathbf{t}|\mathbf{x}_k^a, \boldsymbol{\theta}_a)p(\mathbf{x}_k^a), \tag{4.2}$$

where the conditional distribution $p(\mathbf{t}|\mathbf{x}_k^a, \boldsymbol{\theta}_a)$ is a member of the exponential family given by equation (2.67) and $p(\mathbf{x}_k^a)$ is a uniform prior over the latent space given by equation (2.54).

The mixture can be trained by an EM algorithm. Given the training data points $\zeta = \{\mathbf{t}_1, \ldots, \mathbf{t}_N\}$ of i.i.d. in the data space, the log-likelihood function ($\log \mathcal{L}$) of the mixture of LTMs is

$$\begin{aligned} \log \mathcal{L} = \sum_{n=1}^{N} \log p(\mathbf{t}_n) &= \sum_{n=1}^{N} \log \sum_{a=1}^{A} P(a)p(\mathbf{t}_n|a) \\ &= \sum_{n=1}^{N} \log \sum_{a=1}^{A} P(a) \sum_{k=1}^{K_a} p(\mathbf{t}_n|\mathbf{x}_k^a, \boldsymbol{\theta}_a)p(\mathbf{x}_k^a). \end{aligned} \tag{4.3}$$

Now the missing data include binary assignment variables $\nu_{n,a}$, indicating which model is responsible for generating which data point $\mathbf{t}_n$, and z_{kn}^a indicating which latent space centre $\mathbf{x}_k^a$ in model a generated $\mathbf{t}_n$. Thus the corresponding complete data log-likelihood is

$$E_{comp} = \log \mathcal{L}_{comp} = \sum_{n=1}^{N} \sum_{a=1}^{A} \nu_{n,a} \sum_{k=1}^{K_a} z_{kn}^a \log \left(P(a)p(\mathbf{t}_n, \mathbf{x}_k^a) \right). \tag{4.4}$$

We do not know the values of z_{kn}^a and $\nu_{n,a}$, but use their conditional expectations, which are given by posterior probabilities R_{kn}^a and $P(a|\mathbf{t}_n)$, respectively. R_{kn}^a can be computed from equation (2.72), corresponding to the competition among the latent space centres within each LTM a. $P(a|\mathbf{t}_n)$ measures the probability that the observed value $\mathbf{t}_n$ was generated by the ath component of the mixture, and by using Bayes' theorem it is given by

$$P(a|\mathbf{t}_n) = \frac{P(a)p(\mathbf{t}_n|a)}{\sum_{a'}^{A} P(a')p(\mathbf{t}_n|a')}. \tag{4.5}$$

Considering the corresponding expected complete data log-likelihood, we have

$$\langle E_{comp} \rangle = \sum_{n=1}^{N} \sum_{a=1}^{A} P(a|\mathbf{t}_n) \sum_{k=1}^{K_a} R_{kn}^a \log(P(a)p(\mathbf{t}_n, \mathbf{x}_k^a)). \tag{4.6}$$

Maximisation of (4.6) with respect to $P(a)$, using a Lagrange multiplier, gives the estimation equation of mixture coefficients in the $M-$step:

$$P(a) = \frac{1}{N} \sum_{n=1}^{N} P(a|\mathbf{t}_n). \tag{4.7}$$

Similarly, maximisation of (4.6) with respect to parameters $\boldsymbol{\theta}_a$, gives:

$$\mathbf{T}\boldsymbol{R}_a^T \boldsymbol{\Phi}_a^T = \mathbf{g}(\boldsymbol{\theta}_a \boldsymbol{\Phi}_a) \boldsymbol{G}_a \boldsymbol{\Phi}_a^T, \tag{4.8}$$

where $\boldsymbol{\Phi}_a$ is a $M \times K$ matrix with elements $(\boldsymbol{\Phi}_a)_{kj} = \phi_j(\mathbf{x}_k)$, $\mathbf{T}$ is a data matrix, $\boldsymbol{R}_a$ is a $K \times N$ matrix containing *scaled* responsibilities $(\boldsymbol{R}_a)_{kn} = P(a|\mathbf{t}_n)R_{kn}^a$, and $\boldsymbol{G}_a$ is a $K \times K$ diagonal elements corresponding to responsibilities of latent space centres for the whole data sample, $(\boldsymbol{G}_a)_{kk} = \sum_{n=1}^{N} (\boldsymbol{R}_a)_{kn}$.

4.3 General framework for hierarchical latent trait models

The hierarchical latent trait model arranges a set of LTMs and their corresponding plots in a tree structure $\mathcal{T}$ in a top-down fashion. In this section, we describe how a hierarchical tree is built.

4.3.1 Hierarchical trees

Consider a hierarchical tree $\mathcal{T}$ shown in Figure 4.1. We follow the notation (see Table 4.1) introduced in (Tiňo and Nabney, 2002). The position of each model in $\mathcal{T}$ is given by two numbers, which specify its level and its index in that level. The *Root* is at level 1 of the structure $\mathcal{T}$.

In fact, we can construct a simple two-level hierarchical tree in this way: first, a single LTM is used to visualise the whole dataset in the top level (*Root*), then we consider a mixture of A LTMs to fit the density of the data by applying the method described in section 4.2. The user decides an appropriate number of models to fit at the second level and their initial positions on the basis of the top plot. The user selects points $\mathbf{c}_i$ (e.g., the centre of each cluster). These points are transformed into the data space and then used to initialise parameters of each sub-model. We will describe in more detail how this is done in section 4.3.5. This is an *interactive* way with a practical applicability.

Note that since this is a visualisation method, the quality of which is not particularly "quantifiable" but rather is assessed subjectively by the user, it does make sense to allow the user to choose the initial centres for the clustering procedure. Remember that the user only selects the initialisation, and an EM optimisation of the clusters is still performed at each stage. Perhaps using K-means is another choice, however, we still need to choose K and we need to initialise the centres reasonably too. The user can usually make a good choice of sub-models locations when the plot shows clusters.

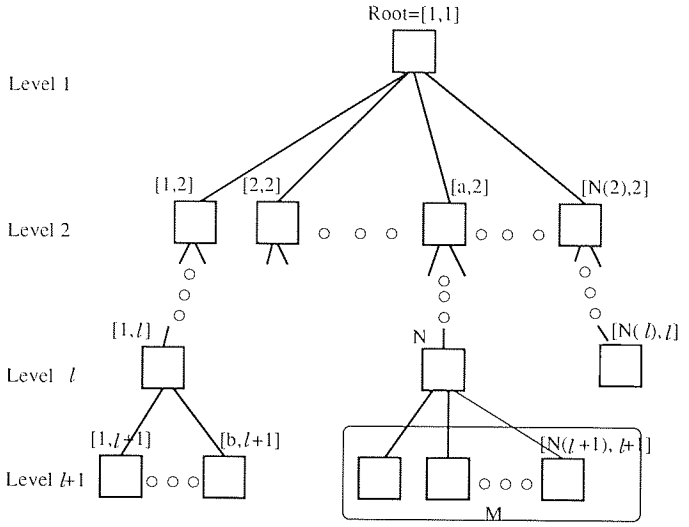


Figure 4.1: An example of a hierarchical tree.

Notation	Explanation	Examples
Parent($\mathcal{M}$)	the first-generation ancestor of $\mathcal{M}$	$Parent([a,2]) = Root$, $Parent([b,\ell+1]) = [1,\ell]$.
Children($\mathcal{M}$)	the set of first-generation descendants of $\mathcal{M}$	$Children(Root) = \{[1,2],[2,2],...,[N(2),2]\}$, $Children([1,\ell]) = \{[1,\ell+1],[2,\ell+1],\cdots,[b,\ell+1]\}$.
Level($\mathcal{M}$)	level of $\mathcal{M}$ in $\mathcal{T}$	$Level(Root) = 1$, $Level([a,2]) = 2$, $Level([b,\ell+1]) = \ell+1$.
Nodes(ℓ)	the set of nodes at level ℓ,	$Nodes(\ell) = \bigcup_{\mathcal{M}\in Nodes(\ell-1)} Children(\mathcal{M})$, $Nodes(1) = \{Root\}$, $Nodes(\ell+1) = \{[1,\ell+1],[2,\ell+1],...,[N(\ell+1),\ell+1]\}$
Path($\mathcal{M}$)	N-tuple of nodes defining the path from $Root$ to $\mathcal{M}$, where $N = Level(\mathcal{M})$	$Path(Root) = (Root)$, $Path([a,2]) = (Root,[a,2])$, $Path([b,\ell+1]) = (Root,[1,2],\cdots,[1,\ell],[b,\ell+1])$.
Leaves(**T**)	the set of nodes without children	$Leaves(\mathcal{T}) = \{[2,2],\cdots,[N(\ell),\ell],\cdots,[N(\ell+1),\ell+1]\}$.

Note: $Path(\mathcal{M})$ can be written in an element-wise form, e.g.,
$Path([b,\ell+1])_1 = Root, Path([b,\ell+1])_2 = [1,2],\cdots,Path([b,\ell+1])_{\ell+1} = [b,\ell+1]$.

Table 4.1: Notation for a hierarchical tree

For example, consider the visualisation of a small text dataset in section 2.6.1. After training a top level LTM, we constructed a mixture of LTMs on four regions of interest centred (shown with indexed numbers) at the four different topics. Each LTM in the mixture was supposed to fit the distribution of the corresponding topic. Figure 4.2 shows data projections in a two-level hierarchy. Four labeled regions are presented at the second level from left to right. Intuitively, it may be considered that we saw the data points in the data space from the different view, so we obtained different appearances of the same data class at different levels.

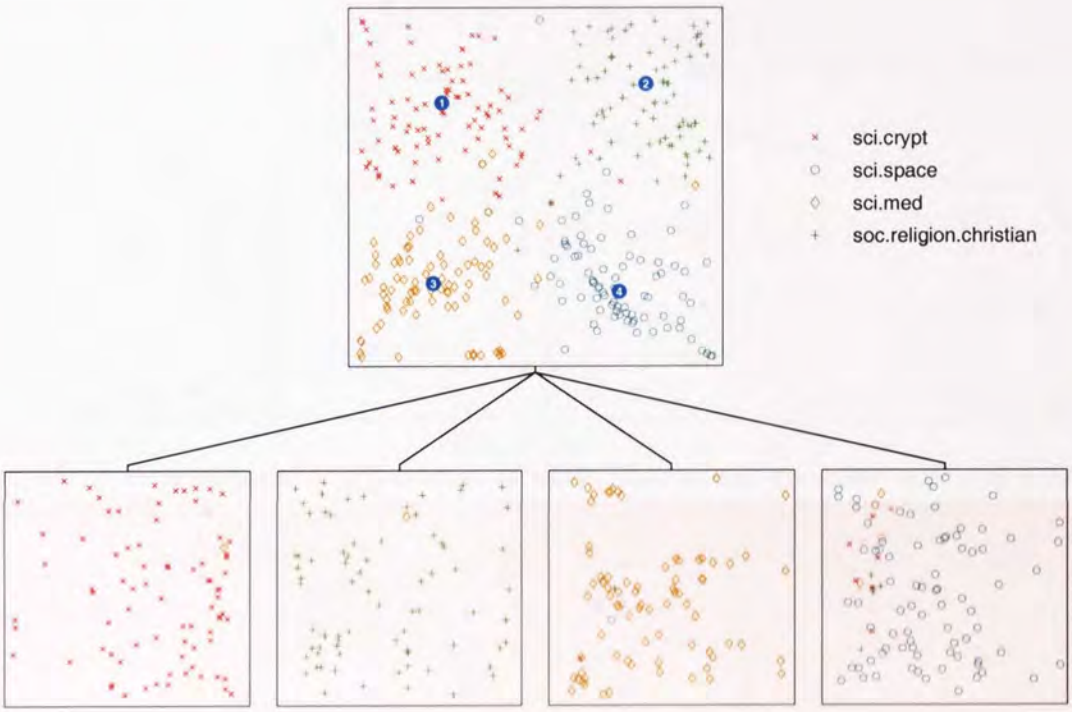


Figure 4.2: A visualisation plot of a mixture of LTMs fitted to the document data.

Figure 4.3 displays a result with randomly generated "centres" of sub-models. For clarity, we also labeled those four centres on the plot. As is clear, projection plots with random centres are much poorer than those in Figure 4.2. For these two mixture models, we plotted their respective priors in Figure 4.4. We see that the second prior of the mixture with random selection is very low. Thus the mixture just worked with the other three models. It suggests that the results of training from randomly generating centers are less robust.

4.3.2 Training the hierarchy of LTMs

To extend the hierarchy to a third level is more complicated, since we need to know which model at the second level and which child of it was responsible for generating $\mathbf{t}_n$, while for a simple two-level hierarchy, we just need to know which model in the second level generated $\mathbf{t}_n$. The whole training procedure of the hierarchical LTMs is like this: first, a *Root* LTM is trained and used to visualise the

Figure 4.3: A visualisation plot of a mixture of LTMs fitted to the document data with randomly initialised models.

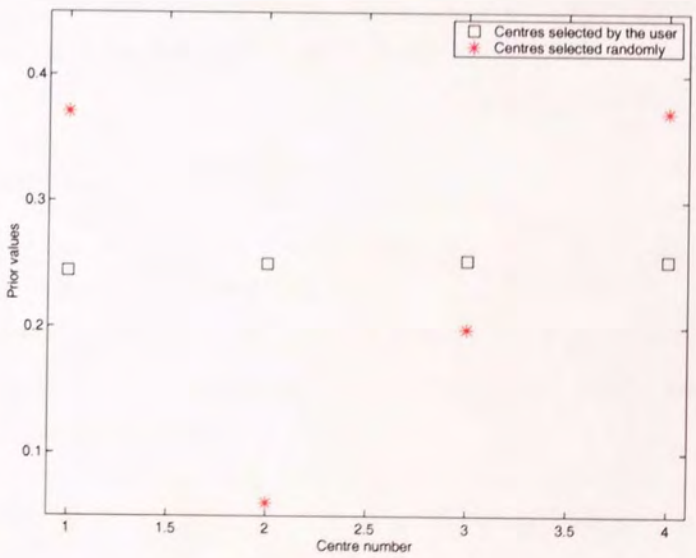


Figure 4.4: Initialised priors on sub-models.

data at the top level; then the user determines regions of interest on the visualisation plot. These regions of interest are then transformed into the data space in order to build a set of child LTMs. After seeing a set of new plots at the second level, the user can decide if it is necessary to continue and if so, they will select further regions in the lower level plots that they would like to model in a greater detail. The training of hierarchy of LTMs proceeds in a recursive fashion.

The hierarchy of LTMs is trained using an EM algorithm to maximise its likelihood with respect to the data sample $\zeta = \{\mathbf{t}_1, \mathbf{t}_2, \cdots, \mathbf{t}_N\}$. The log-likelihood function of the hierarchy $\mathcal{T}$ of LTMs is

$$\log \mathcal{L} = \sum_{n=1}^{N} \log p(\mathbf{t}_n|\mathcal{T}), \tag{4.9}$$

where the distribution $p(\mathbf{t}_n|\mathcal{T})$ is a mixture of models $\mathcal{M}$ at leaves of the tree $\mathcal{T}$. Each model $\mathcal{M}$ defines a probability distribution $p(\mathbf{t}|\mathcal{M})$ in the data space. We have rewritten equation (4.1) for a hierarchical mixture as

$$p(\mathbf{t}|\mathcal{T}) = \sum_{\mathcal{M} \in Leaves(\mathcal{T})} P(\mathcal{M}) p(\mathbf{t}|\mathcal{M}), \tag{4.10}$$

where unconditional mixing coefficients $P(\mathcal{M})$ are given by

$$\begin{aligned} P(\mathcal{M}) &= \prod_{i=2}^{Level(\mathcal{M})} P(Path(\mathcal{M})_i | Path(\mathcal{M})_{i-1}) \\ &= P(\mathcal{M}|Parent(\mathcal{M}))P(Parent(\mathcal{M})), \end{aligned} \tag{4.11}$$

and $P(\mathcal{M}|Parent(\mathcal{M}))$ is called parent-conditional mixture coefficient. For *Root*, $P(Root) = 1$.

Note that models corresponding to non-leaf nodes of $\mathcal{T}$ play their role only in the process of creating the hierarchical model. Once the hierarchy is trained and the mixture coefficients (4.11) are established, we need these non-leaf models only if we wish to extend or retrain the hierarchical model structure in the future (Tiňo and Nabney, 2002). Thus to extend the hierarchy to level $\ell + 1$, we write $p(\mathbf{t}|\mathcal{T})$ in two parts: one is given by leaves which do not belong to level $\ell + 1$, and the other by leaves at level $\ell + 1$,

$$\begin{aligned} p(\mathbf{t}|\mathcal{T}) &= \sum_{\mathcal{M} \in Leaves(\mathcal{T})} P(\mathcal{M}) p(\mathbf{t}|\mathcal{M}) \\ &= \sum_{\mathcal{M} \in Leaves(\mathcal{T}) \setminus Nodes(\ell+1)} P(\mathcal{M}) p(\mathbf{t}|\mathcal{M}) + \sum_{\mathcal{M} \in Nodes(\ell+1)} P(\mathcal{M}) p(\mathbf{t}|\mathcal{M}). \end{aligned} \tag{4.12}$$

Since at present, models $\mathcal{M}$ at level $\ell + 1$ are new, while all the other nodes in the hierarchy are fixed, the log-likelihood function ($\log \mathcal{L}$) is maximised by maximising the *restricted log-likelihood function* (E) confined only to the LTMs at level $\ell + 1$,

$$E^{(\ell+1)} = \sum_{n=1}^{N} \log \left[\sum_{\mathcal{M} \in Nodes(\ell+1)} P(\mathcal{M}) p(\mathbf{t}_n|\mathcal{M}) \right]. \tag{4.13}$$

Using equation (4.11), equation (4.13) can be written as

$$E^{(\ell+1)} = \sum_{n=1}^{N} \log \left[\sum_{\mathcal{M} \in Nodes(\ell+1)} P(\mathcal{M}|Parent(\mathcal{M}))P(Parent(\mathcal{M}))p(\mathbf{t}_n|\mathcal{M}) \right]. \tag{4.14}$$

Determination of the parameters of the models can again be viewed as a missing data problem. Here, in contrast with a mixture of latent trait models (see section 4.2), we have three types of hidden variable. We denote $\mathcal{N}$ is a node at level ℓ.

Assignment variables $\nu_{n,\mathcal{N}}$. These indicate which LTM at level ℓ generated the nth data point. Equation (4.14) can be rewritten as

$$E^{(\ell+1)} = \sum_{n=1}^{N} \sum_{\mathcal{N} \in Nodes(\ell)} \nu_{n,\mathcal{N}} \log \left[\sum_{\mathcal{M} \in Children(\mathcal{N})} P(\mathcal{M}|\mathcal{N}) P(\mathcal{N}) p(\mathbf{t}_n|\mathcal{M}) \right]. \tag{4.15}$$

We do not know the value of $\nu_{n,\mathcal{N}}$, but can compute its expectation, which is the posterior probability $P(\mathcal{N}|\mathbf{t}_n)$ that LTM $\mathcal{N}$ generated $\mathbf{t}_n$. We take the expectation of (4.15) and obtain

$$\langle E^{(\ell+1)} \rangle = \sum_{n=1}^{N} \sum_{\mathcal{N} \in Nodes(\ell)} P(\mathcal{N}|\mathbf{t}_n) \log \left[P(\mathcal{N}) \sum_{\mathcal{M} \in Children(\mathcal{N})} P(\mathcal{M}|\mathcal{N}) p(\mathbf{t}_n|\mathcal{M}) \right]. \tag{4.16}$$

Assignment variables $\nu_{n,\mathcal{M}|\mathcal{N}}$. These indicate that, given the parent $\mathcal{N}$ responsible for generating a point $\mathbf{t}_n$, which of its children $\mathcal{M}$ generated $\mathbf{t}_n$. Again, we are able to calculate the expectation of $\nu_{n,\mathcal{M}|\mathcal{N}}$, which is given by (parent-conditional) responsibilities $P(\mathcal{M}|\mathcal{N}, \mathbf{t}_n)$.

Assignment variables $z_{kn}^{\mathcal{M}}$. These indicate which latent space center $\mathbf{x}_k^{\mathcal{M}} \in \mathcal{H}, k = 1, 2, \ldots, K$ of the LTM $\mathcal{M}$ corresponds to the noise model that generated $\mathbf{t}_n$ (see equation (2.67)). As before, we only have the responsibilities $R_{kn}^{\mathcal{M}}$ given by equation (2.72).

Now we are ready to write the complete-data restricted likelihood function confined only to the LTMs at level $\ell + 1$. From equation (4.16), it has the form

$$E_{comp}^{(\ell+1)} = \sum_{n=1}^{N} \sum_{\mathcal{N} \in Nodes(\ell)} P(\mathcal{N}|\mathbf{t}_n) \sum_{\mathcal{M} \in Children(\mathcal{N})} \nu_{n,\mathcal{M}|\mathcal{N}} \sum_{k=1}^{K_{\mathcal{M}}} z_{kn}^{\mathcal{M}} \log \left[P(\mathcal{N}) P(\mathcal{M}|\mathcal{N}) p(\mathbf{t}_n, \mathbf{x}_k^{\mathcal{M}}) \right]. \tag{4.17}$$

Taking the expectation, we have:

$$\begin{aligned} \langle E_{comp}^{\ell+1} \rangle = & \sum_{n=1}^{N} \sum_{\mathcal{N} \in Nodes(\ell)} P(\mathcal{N}|\mathbf{t}_n) \sum_{\mathcal{M} \in Children(\mathcal{N})} P(\mathcal{M}|\mathcal{N}, \mathbf{t}_n) \\ & \sum_{k=1}^{K_{\mathcal{M}}} R_{kn}^{\mathcal{M}} \log \left[P(\mathcal{N}) P(\mathcal{M}|\mathcal{N}) p(\mathbf{t}_n, \mathbf{x}_k^{\mathcal{M}}) \right]. \end{aligned} \tag{4.18}$$

Since $p(\mathbf{t}_n, \mathbf{x}_k^{\mathcal{M}}) = p(\mathbf{t}_n|\mathbf{x}_k^{\mathcal{M}}, \boldsymbol{\theta}_{\mathcal{M}}) p(\mathbf{x}_k^{\mathcal{M}})$, equation (4.18) involves four terms,

$$\text{term1} = \sum_{n=1}^{N} \sum_{\mathcal{N} \in Nodes(\ell)} P(\mathcal{N}|\mathbf{t}_n) \sum_{\mathcal{M} \in Children(\mathcal{N})} P(\mathcal{M}|\mathcal{N}, \mathbf{t}_n) \sum_{k=1}^{K_{\mathcal{M}}} R_{kn}^{\mathcal{M}} \log P(\mathcal{N}),$$

$$\text{term2} = \sum_{n=1}^{N} \sum_{\mathcal{N} \in Nodes(\ell)} P(\mathcal{N}|\mathbf{t}_n) \sum_{\mathcal{M} \in Children(\mathcal{N})} P(\mathcal{M}|\mathcal{N}, \mathbf{t}_n) \sum_{k=1}^{K_{\mathcal{M}}} R_{kn}^{\mathcal{M}} \log P(\mathcal{M}|\mathcal{N}),$$

$$\text{term3} = \sum_{n=1}^{N} \sum_{\mathcal{N} \in Nodes(\ell)} P(\mathcal{N}|\mathbf{t}_n) \sum_{\mathcal{M} \in Children(\mathcal{N})} P(\mathcal{M}|\mathcal{N}, \mathbf{t}_n) \sum_{k=1}^{K_{\mathcal{M}}} R_{kn}^{\mathcal{M}} \log p(\mathbf{t}_n|\mathbf{x}_k^{\mathcal{M}}, \boldsymbol{\theta}_{\mathcal{M}}),$$

$$\text{term4} = \sum_{n=1}^{N} \sum_{\mathcal{N} \in Nodes(\ell)} P(\mathcal{N}|\mathbf{t}_n) \sum_{\mathcal{M} \in Children(\mathcal{N})} P(\mathcal{M}|\mathcal{N}, \mathbf{t}_n) \sum_{k=1}^{K_{\mathcal{M}}} R_{kn}^{\mathcal{M}} \log p(\mathbf{x}_k^{\mathcal{M}}).$$

With conditions

$$\sum_{k=1}^{K_{\mathcal{M}}} R_{kn}^{\mathcal{M}} = 1, \quad R_{kn}^{\mathcal{M}} \geq 0,$$

and

$$\sum_{\mathcal{M} \in Children(\mathcal{N})} P(\mathcal{M}|\mathcal{N}, \mathbf{t}_n) = 1, \quad P(\mathcal{M}|\mathcal{N}, \mathbf{t}_n) \geq 0,$$

these four terms can be simplified as:

$$\text{term1} = \sum_{n=1}^{N} \sum_{\mathcal{N} \in Nodes(\ell)} P(\mathcal{N}|\mathbf{t}_n) \log P(\mathcal{N}), \tag{4.19}$$

$$\text{term2} = \sum_{n=1}^{N} \sum_{\mathcal{N} \in Nodes(\ell)} P(\mathcal{N}|\mathbf{t}_n) \sum_{\mathcal{M} \in Children(\mathcal{N})} P(\mathcal{M}|\mathcal{N}, \mathbf{t}_n) \log P(\mathcal{M}|\mathcal{N}), \tag{4.20}$$

$$\text{term3} = \sum_{n=1}^{N} \sum_{\mathcal{N} \in Nodes(\ell)} P(\mathcal{N}|\mathbf{t}_n) \sum_{\mathcal{M} \in Children(\mathcal{N})} P(\mathcal{M}|\mathcal{N}, \mathbf{t}_n) \sum_{k=1}^{K_{\mathcal{M}}} R_{kn}^{\mathcal{M}} \log p(\mathbf{t}_n|\mathbf{x}_k^{\mathcal{M}}, \boldsymbol{\theta}_{\mathcal{M}}), \tag{4.21}$$

$$\text{term4} = \log \frac{1}{K_{\mathcal{M}}} \sum_{n=1}^{N} \sum_{\mathcal{N} \in Nodes(\ell)} P(\mathcal{N}|\mathbf{t}_n). \tag{4.22}$$

Term1 and term4 are constant with respect to the adjustable parameters of LTMs at level $\ell + 1$. Maximising (4.20) with respect to the parent-conditional mixture coefficients $P(\mathcal{M}|\mathcal{N})$, must take account of the constraint

$$\sum_{\mathcal{M} \in Children(\mathcal{N})} P(\mathcal{M}|\mathcal{N}) = 1.$$

After a straightforward calculation (see Appendix C), we obtain

$$P(\mathcal{M}|Parent(\mathcal{M})) = \frac{\sum_{n=1}^{N} P(\mathcal{M}|\mathbf{t}_n)}{\sum_{n=1}^{N} P(Parent(\mathcal{M})|\mathbf{t}_n)}, \tag{4.23}$$

where

$$P(\mathcal{M}|\mathbf{t}_n) = P(\mathcal{M}|Parent(\mathcal{M}), \mathbf{t}_n) P(Parent(\mathcal{M})|\mathbf{t}_n), \tag{4.24}$$

$$P(\mathcal{M}|Parent(\mathcal{M}), \mathbf{t}_n) = \frac{P(\mathcal{M}|Parent(\mathcal{M}))p(\mathbf{t}_n|\mathcal{M})}{\sum_{\mathcal{M}' \in [\mathcal{M}]} P(\mathcal{M}'|Parent\mathcal{M})p(\mathbf{t}_n|\mathcal{M}')}, \tag{4.25}$$

and

$$[\mathcal{M}] = Children(Parent(\mathcal{M})). \tag{4.26}$$

Maximising (4.21) with respect to $\boldsymbol{\theta}_{\mathcal{M}}$, using equations (2.67) and (2.69), we obtain

$$\mathbf{T} \boldsymbol{R}_{\mathcal{M}}^T \boldsymbol{\Phi}_{\mathcal{M}}^T = \mathrm{g}(\boldsymbol{\theta}_{\mathcal{M}} \boldsymbol{\Phi}_{\mathcal{M}}) \boldsymbol{G}_{\mathcal{M}} \boldsymbol{\Phi}_{\mathcal{M}}^T, \tag{4.27}$$

where $\boldsymbol{R}_{\mathcal{M}} = (r_{kn}^{\mathcal{M}})_{k=1,\ldots,K, n=1,\ldots,N}$. $r_{kn}^{\mathcal{M}}$ is calculated by using the equation (2.72) to calculate $R_{kn}^{\mathcal{M}}$ firstly, then rescaled by $P(\mathcal{M}|\mathbf{t}_n)$ from equation (4.24), i.e. $r_{kn}^{\mathcal{M}} = P(\mathcal{M}|\mathbf{t}_n) R_{kn}^{\mathcal{M}}$; $\boldsymbol{G}_{\mathcal{M}}$ is a diagonal matrix with elements $(G_{\mathcal{M}})_{kk} = \sum_{n=1}^{N} r_{kn}^{\mathcal{M}}$.

When solving (4.27), in general a non-linear optimization algorithm is required (see section 2.6.1). But if the link function $\mathbf{g}(\cdot)$ is the identity, one gets the closed form for updating $\mathbf{W}_{\mathcal{M}}$ in $M-$step of HGTM, while $\beta_{\mathcal{M}}^{-1}$ must be calculated by using the following equation (Tiňo and Nabney, 2002):

$$\frac{1}{\beta_{\mathcal{M}}} = \frac{\sum_{n=1}^{N} P(\mathcal{M}|\mathbf{t}_n) \sum_{k=1}^{K_{\mathcal{M}}} R_{kn}^{\mathcal{M}} ||\mathbf{W}_{\mathcal{M}} \boldsymbol{\phi}(\mathbf{x}_k^{\mathcal{M}}) - \mathbf{t}_n||^2}{D \sum_{n=1}^{N} P(\mathcal{M}|\mathbf{t}_n)}. \tag{4.28}$$

4.3.3 Summary of the EM algorithm

The hierarchical LTM is trained using EM to maximise its likelihood with respect to the data sample $\zeta = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_N\}$. The hierarchy is trained in a top-down fashion, starting with the *Root* model, and proceeds in a recursive fashion. When child LTMs of a node are being constructed, the EM algorithm calculates expectations of assignment variables in the $E-$step and updates the corresponding parameters in the $M-$step until it converges.

$E-$step

We estimate the posterior over all hidden variables, using the "old" values of LTM parameters. Given a data point $\mathbf{t}_n \in \mathcal{D}$,

- imposing $P(Root|\mathbf{t}_n) = 1$, the unconditional (on parent) model responsibilities are recursively determined by (4.24);
- equation (4.25) is used to compute the model responsibilities corresponding to the competition among models belonging to the same parent;
- responsibilities of the latent space centres $\mathbf{x}_k^{\mathcal{M}}$, $k = 1, 2, ..., K_{\mathcal{M}}$, corresponding to the competition among the latent space centres in each model $\mathcal{M}$ are calculated using (2.72).

$M-$step

The parameters are estimated using the posterior over hidden variables computed in the $E-$step.

- Parent-conditional mixture coefficients are determined by equation (4.23).
- Parameters $\boldsymbol{\theta}_{\mathcal{M}}$ are calculated by solving equation (4.27). For HGTM, inverse variances $\beta_{\mathcal{M}}^{-1}$ is computed using (4.28).

4.3.4 Mixed data

Since the observed variables are assumed to be independent given the latent variables, we can cope with mixed data by simply multiplying the corresponding distribution in the $E-$step. In the $M-$step, the formulation for a Gaussian noise model is the same as the GTM, but for Bernoulli and multinomial distributions, we must use those described in the LTM (see section 2.6.1).

4.3.5 Practical Considerations

Initialisation

As shown in Figure 4.3, with an unsuccessful initialisation, the mixture will have a poor performance. The basic idea of our initialisation procedure is illustrated in Figure 4.5. When initialising sub-models there are two things to determine: the number of sub-models and the initial parameters of the sub-models. We view the problem of initialising sub-model parameters primarily as one of locating which region of data space each sub-model should be responsible for. To do this, regions of interest are defined by the user in the latent (visualisation) space. The points $\mathbf{c}_i$ selected in the latent space $\mathcal{H}$ correspond to the "centres" of these regions. $\mathbf{c}_i$ is mapped to $\mathbf{\Omega}_{\mathcal{N}}(\mathbf{c}_i)$, the image under the corresponding parent LTM $\mathcal{N}$,

$$\mathbf{\Omega}_{\mathcal{N}}(\mathbf{c}_i) = g(\boldsymbol{\theta}_{\mathcal{N}}\mathbf{\Phi}(\mathbf{c}_i)).$$

The "regions of interest" in data space are created as Voronoi compartments $V_i, i = 1, \ldots, A$ (Aurenhammer, 1991):

$$V_i = \left\{ \mathbf{t} \in \mathcal{R}^D | d(\mathbf{t}, \mathbf{\Omega}_{\mathcal{N}}(\mathbf{c}_i)) = \min_j d(\mathbf{t}, \mathbf{\Omega}_{\mathcal{N}}(\mathbf{c}_j)) \right\}, \tag{4.29}$$

which are bounded by hyperplanes. Each compartment contains the corresponding mapped centre $\mathbf{\Omega}_{\mathcal{N}}(\mathbf{c}_i) \in \mathcal{D}$.

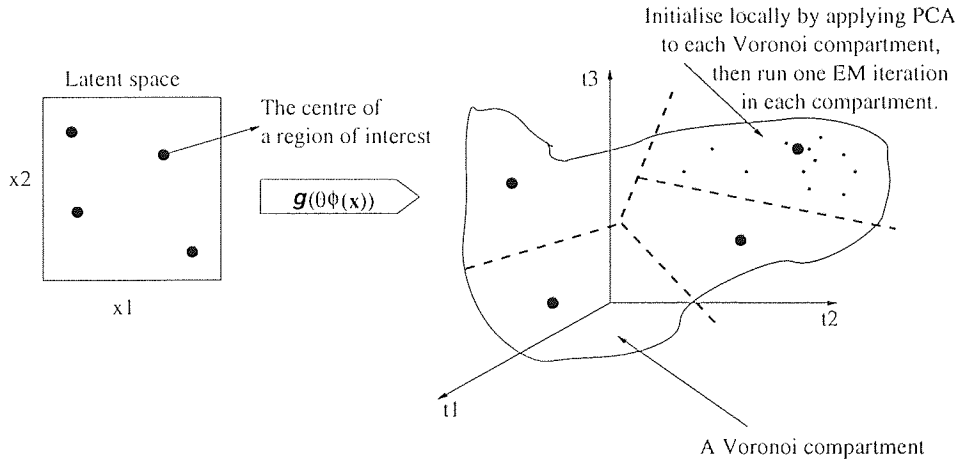


Figure 4.5: Initialisation by applying PCA locally.

In the case of a Gaussian noise model, the child LTMs are initialised by local PCA in the corresponding Voronoi compartments (Tiňo and Nabney, 2002). When using other noise models such as Bernoulli or multinomial distributions, the PCA-initialised LTMs are in addition individually trained for one EM iteration only on the data in the corresponding Voronoi compartment. This is different from fitting the mixture of child models to the whole dataset. This localised EM iteration "settles" the component LTMs to their corresponding modelling regions. Empirically, this initialisation strategy works very well.

For example, we trained a hierarchical tree down to level three on the small document dataset. Figure 4.6 shows the result obtained with performing the additional initialisation step. We see at the third level four topics are separated well. Figure 4.7 displays the result obtained without the

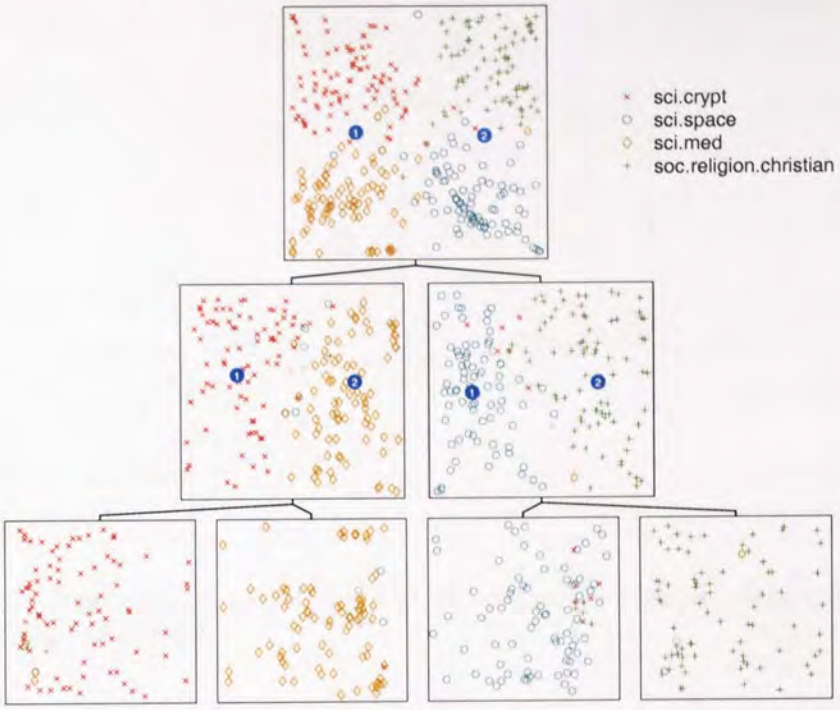


Figure 4.6: A visualisation plot of a hierarchy of LTMs fitted on the document data with a localised EM iteration in the initialisation.

additional initialisation step. In the latter, the third level-three model "swallowed up" nearly half the data points belonging to the fourth model of the same level. It suggests that the PCA initialisation alone does not "match" the discrete distribution well.

After the initialisation, the full hierarchical training procedure described in section 4.3.3 is used.

Setting a threshold for the parental responsibility

For each child mixture in the hierarchy, if they are fitted to the whole dataset, the responsibilities of its parent model for many data points will approach zero. This implies that the weighted responsibilities for the components of the mixture will be at least as small. So we need only train the children mixture to a *reduced* dataset. A threshold is set to discard those points whose parental responsibility is less than this value. We adopted a threshold $\epsilon = 10^{-5}$ for the experiments within this thesis, and observed a considerable computational advantage, particular at lower levels in the hierarchy.

4.4 The hierarchical LTM visualisation implementation

Note that in this chapter, HLTM is used in an interactive mode, where the user is responsible for choosing the number of components (of a mixture) and their corresponding positions. We have

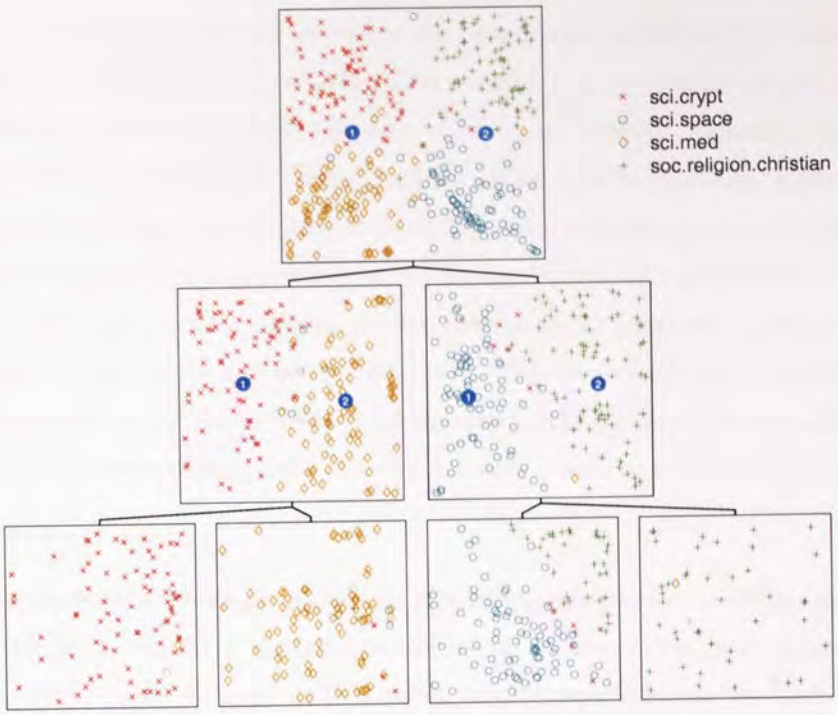


Figure 4.7: A visualisation plot of a hierarchy of LTMs fitted on the document data without a localised EM iteration in the initialisation.

developed an HLTM software system using the NETLAB toolbox, which is available from the URL `http://www.ncrg.aston.ac.uk/netlab/`. To visualise the data, we plot the posterior mean of each data point in the latent (visualisation) space. We first generate a single top-level latent variable model and plot it. The user then selects "centres" of regions of interest. This is done by simply clicking points in the latent (visualisation) space. These "centres" are displayed as circles labeled by numbers. When the plots corresponding to a hierarchy of LTMs are shown, they are presented in an order which is determined by the labels of the "centres". The whole visualisation implementation involves five functions: **(1)** to show data projections in a hierarchical visualisation plot; **(2)** to exhibit child-modulated ancestor plots; **(3)** to list a group of points which are closest to a point chosen by the user; **(4)** to display a corresponding hierarchical visualisation of magnification factors; **(5)** to present a corresponding hierarchical visualisation of local curvatures (GTMs only).

- data projections in a hierarchical visualisation plot

  The child plots from left to right correspond to the index a from 1 to A. As in (Bishop and Tipping, 1998) and (Tiňo and Nabney, 2002), we adopt the strategy that the intensity of a data point $\mathbf{t}_n$ on each plot is modified by the responsibility $P(\mathcal{M}|\mathbf{t}_n)$ (see equations (4.24), (4.25) and (4.26)). If one model takes most of the responsibility for a particular data point, this point will be seen clearly only on the corresponding plot, while those are not well captured by a particular plot will appear with low intensity.

- child-modulated ancestor plots

The user can visualise the points captured by a particular child LTM $\mathcal{M}$, by modifying the plot of its parent, $Parent(\mathcal{M})$, so that instead of the parent responsibilities, $P(Parent(\mathcal{M})|\ \mathbf{t}_n)$, the responsibilities of the model $\mathcal{M}$, $P(\mathcal{M}|\ \mathbf{t}_n)$, are used. This is done by simply clicking with a mouse on a child LTM plot which the user is willing to observe. Alternatively, the user can modulate with responsibilities $P(Parent(\mathcal{M})|\ \mathbf{t}_n)$ all the ancestor plots up to *Root*, i.e. all plots appearing in $Path(Parent(\mathcal{M}))$. The chosen child plot is highlighted by a bold red frame. The ancestor plots appear in bold green frames. Such a modulation of ancestor plots is an important tool helping the user to relate children plots to their parents. This way we can see which points in ancestor plots are explained in the child plot. This is a useful tool especially when high curvatures happen in the parent plot, in which case points far from a region centre $\boldsymbol{c}_i$ may be explained in the corresponding child plot. An example can be seen in section 4.5.3.

- a list of a group of points

  The user clicks with a mouse on a point in the visualisation space to obtain more knowledge on that data point. We list a group of points, closest to the clicked point in the latent space, with their indices in the dataset, or ID number (e.g. compounds may be identified by labels) directly if available. According to these indices (IDs), one can find out property values of those points in the data space. This can be of help to further research why projections of these points are so close and enable the user to identify those properties in common among points in a given cluster.

- corresponding hierarchical visualisation of magnification factors

  The hierarchical structure of plots used for plotting the LTMs' projections is also used to show the magnification factors of LTMs in the hierarchy. For every LTM, the method for evaluating magnification factors is the same as described in section 2.6.2. The intensities of the magnification factors are scaled with respect to the minimal and maximal magnification factors in the whole hierarchy. The scale is shown as a color bar near the top visualisation plot corresponding to the root LTM. The user can get a locally scaled plot of magnification factors by clicking on a chosen plot corresponding to a local LTM $\mathcal{M}$. Magnification factors of the LTM $\mathcal{M}$ are then shown scaled with respect to the minimal and maximal magnification factors of $\mathcal{M}$. A $\log_2$ scale is used for viewing, where values more than 0 indicate the manifold is stretched in data space, and the reverse for compression.

- corresponding local curvatures in the hierarchy for the Gaussian noise model

  For each GTM in the hierarchy tree, the approach for calculating the local curvature is described in section 3.2.1. The basic method for displaying the local directional curvatures of each GTM is the same as described in section 3.2.2. As in the case of magnification factors, the intensity of curvatures in the hierarchy of GTMs is scaled by the minimal and maximal curvatures found in the whole hierarchy. A locally scaled plot of curvatures can be obtained by clicking on a chosen

plot corresponding to a local GTM.

4.5 Experimental results

In this section, we evaluate the hierarchical LTM visualisation algorithm on 3 real-world datasets. We used a common configuration for all models in the hierarchy reported here, though the algorithm is derived in a general setting in which individual LTMs $\mathcal{M}$ in the hierarchy can have different sets of latent space centres $\mathbf{x}_k^{\mathcal{M}}$, $k = 1, 2, ..., K_{\mathcal{M}}$, and basis functions ϕ_j, $j = 1, 2, ..., M_{\mathcal{M}}$. In particular, the latent space $\mathcal{H}$ was taken to be the two-dimensional interval $\mathcal{H} = [-1, 1] \times [-1, 1]$, the latent space centres $\mathbf{x}_k^{\mathcal{M}} \in \mathcal{H}$ were positioned on a regular 15×15 square grid and there were 16 radial basis functions ϕ_j centered on a regular 4×4 square grid. The basis functions were spherical Gaussians with a common width σ chosen equal to the distance from each centre to its neighbour. We account for a bias term by using an additional constant basis function $\phi_{17}(\mathbf{x}) = 1$, for all $\mathbf{x} \in \mathcal{H}$. In all our experiments, we imposed (as usual in LTM) a uniform prior $P(\mathbf{x}_k)$ over the latent space grid. This ensures that all regions of the latent space can be used for visualisation purposes with "equal importance".

Note that, as mentioned in section 4.4, in the interactive mode, the "centres" of the regions of interest are shown as circles labeled by numbers. These numbers determine the order of the corresponding child LTM subplots from left to right. In this case, those plots without labeled numbers are leaves in a hierarchy tree.

4.5.1 Document dataset

As the first example, we tested our algorithm on a binary dataset, which is a subset of the document dataset (see section 1.5.4). 100 data points were randomly selected from each class.

A hierarchy of LTMs down to level four was trained on this data collection and a final hierarchical projection plot is shown in Figure 4.8. The corresponding magnification factor plot is presented in Figure 4.9. By clicking on the last level-three LTM $\mathcal{M}$ modeling points from topic "talk.politics.mideast", we can trace the position of points locally captured by $\mathcal{M}$ in the visualisation plots of all its ancestors. The corresponding plot, Figure 4.10, is a child-modulated ancestor plot (see section 4.4). With reference to Figure 4.8, we see that the position of these points are reasonable.

Interestingly, looking at the fourth level-two LTM $\mathcal{M}$ in Figures 4.8 and 4.9, we found that the points were grouped into two clusters, even though they were from the same topic "talk.politics.mideast". To investigate further, we focused on the corresponding LTM $\mathcal{M}$ model. An enlarged view of the magnification factor plot is presented in Figure 4.11. It indicates that there are two sub-classes separated along the dark boundary appearing in Figure 4.11. The lists of five most probable dictionary words, the top of the reference vector $\mathbf{g}(\boldsymbol{\theta}\boldsymbol{\Phi}(\mathbf{x}_k)$, for each latent space centre is displayed in Figure 4.12. In this plot, each cell of the 15 by 15 table corresponds to one grid point. For the ease of the reader, we have plotted two smaller shaded areas surrounded by dotted-lines. The left one seems likely to be

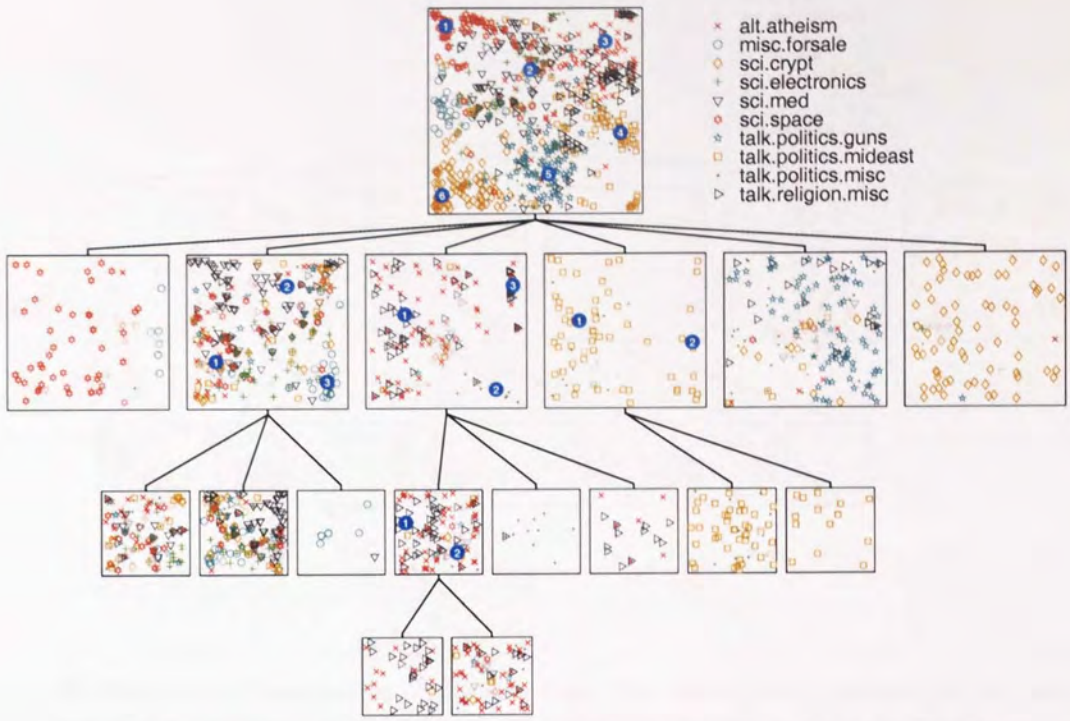


Figure 4.8: A visualisation plot of a hierarchy of LTMs fitted to the document data.

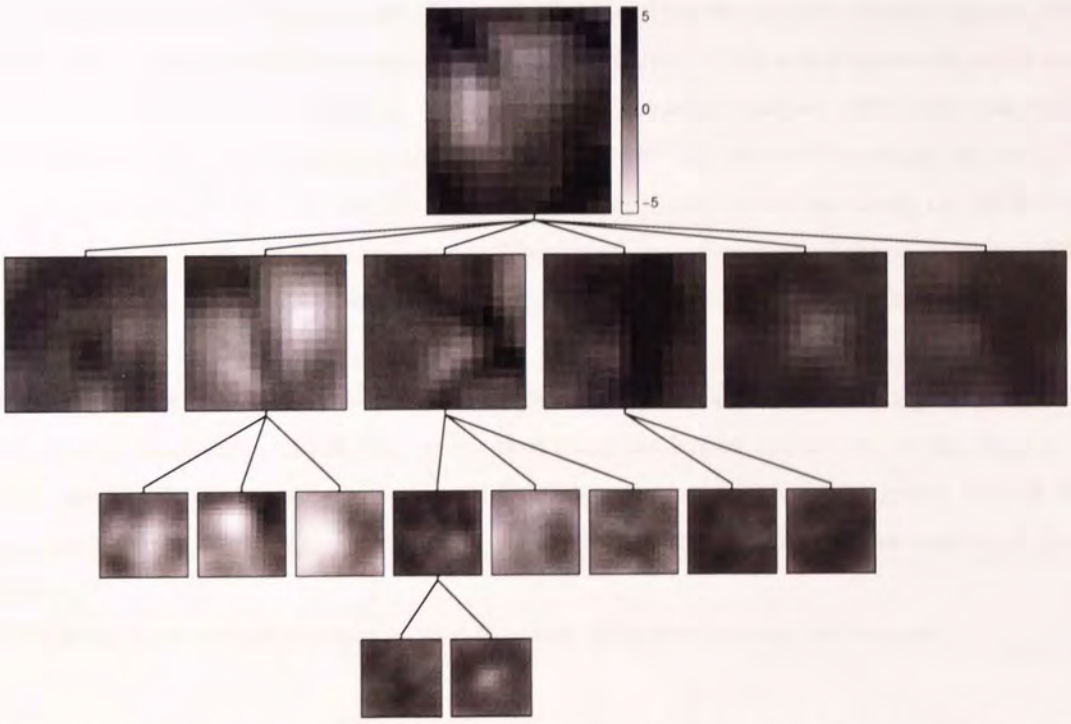


Figure 4.9: A magnification factor plot of a hierarchy of LTMs fitted to the document data.

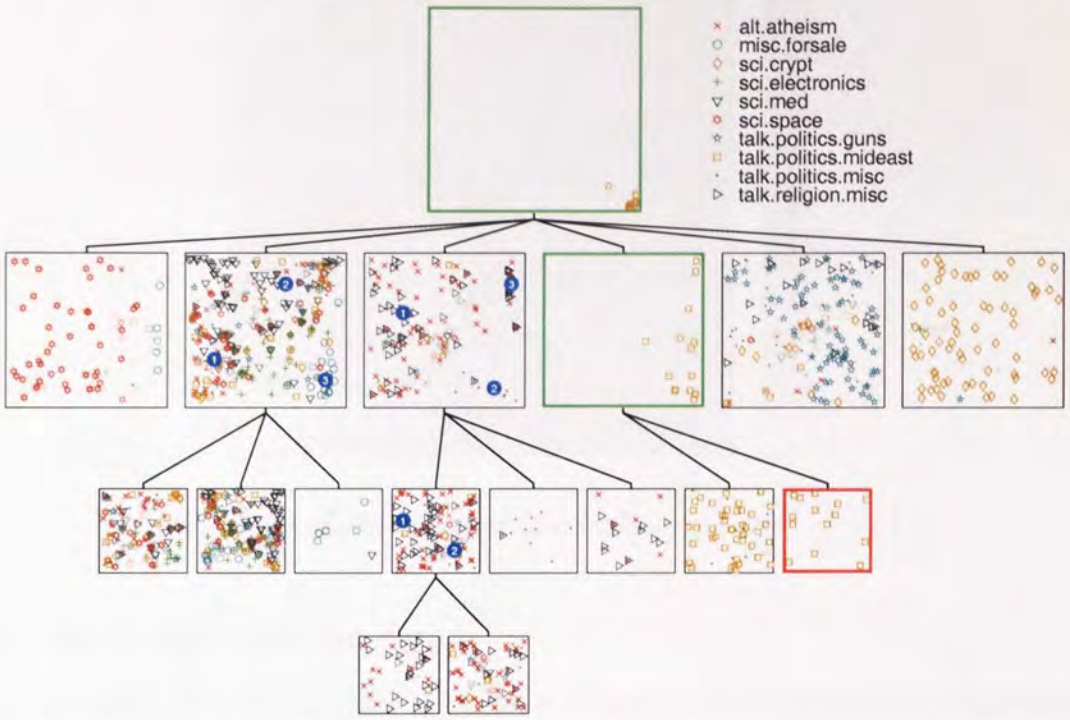


Figure 4.10: Hierarchical visualisation of the text data. The set of points captured by the last LTM at level three of the hierarchy is highlighted in the visualisation plots of all its ancestors.

associated with articles concerning the problem of Israel occupation, while key words inside the right one appear to refer to documents relating to relationships between Armenia, Turkey and the Soviet union. Moreover, we plotted the border of the middle area, between the two shaded regions, using a solid line. We can observe that grid points that lie in the middle of this area store words which are the most probable appearing; those lying in the boundary of this area store key words that can easily be found in different topics. For example, the key word "occupi" can be seen frequently in the junction of the left shaded region and the middle area, and "serdar"[1] can be viewed either on the boundary of the middle area or the right shaded region. These results suggest magnification plots can help us interpret the results from corresponding visualisation plots. Furthermore, they are useful to decide the positions of sub-models.

Figure 4.13 shows a hierarchical projection plot on the same dataset without the additional initialisation step. Looking at the figure, we see that some models at the deeper levels died out. For example, the fourth level-three and first level-four plots do not capture the cluster showed in the corresponding parent plots. It confirms that a simple PCA initialisation is not enough for a discrete distribution.

[1]Note that key words were pre-processed by word-stemming. Thus some words are not complete.

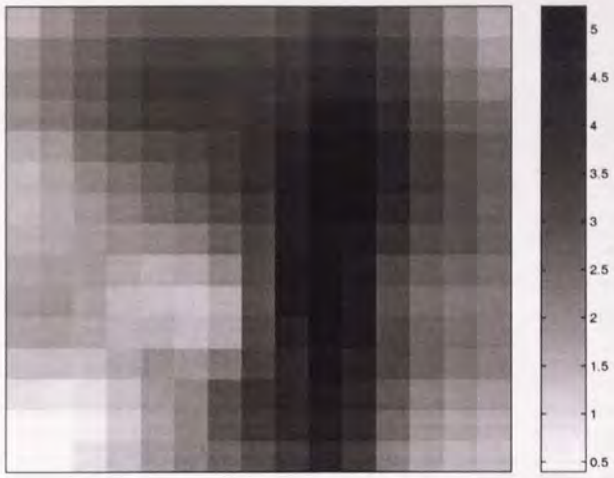


Figure 4.11: A visualisation plot of magnification factors for a LTM

4.5.2 Image segmentation data

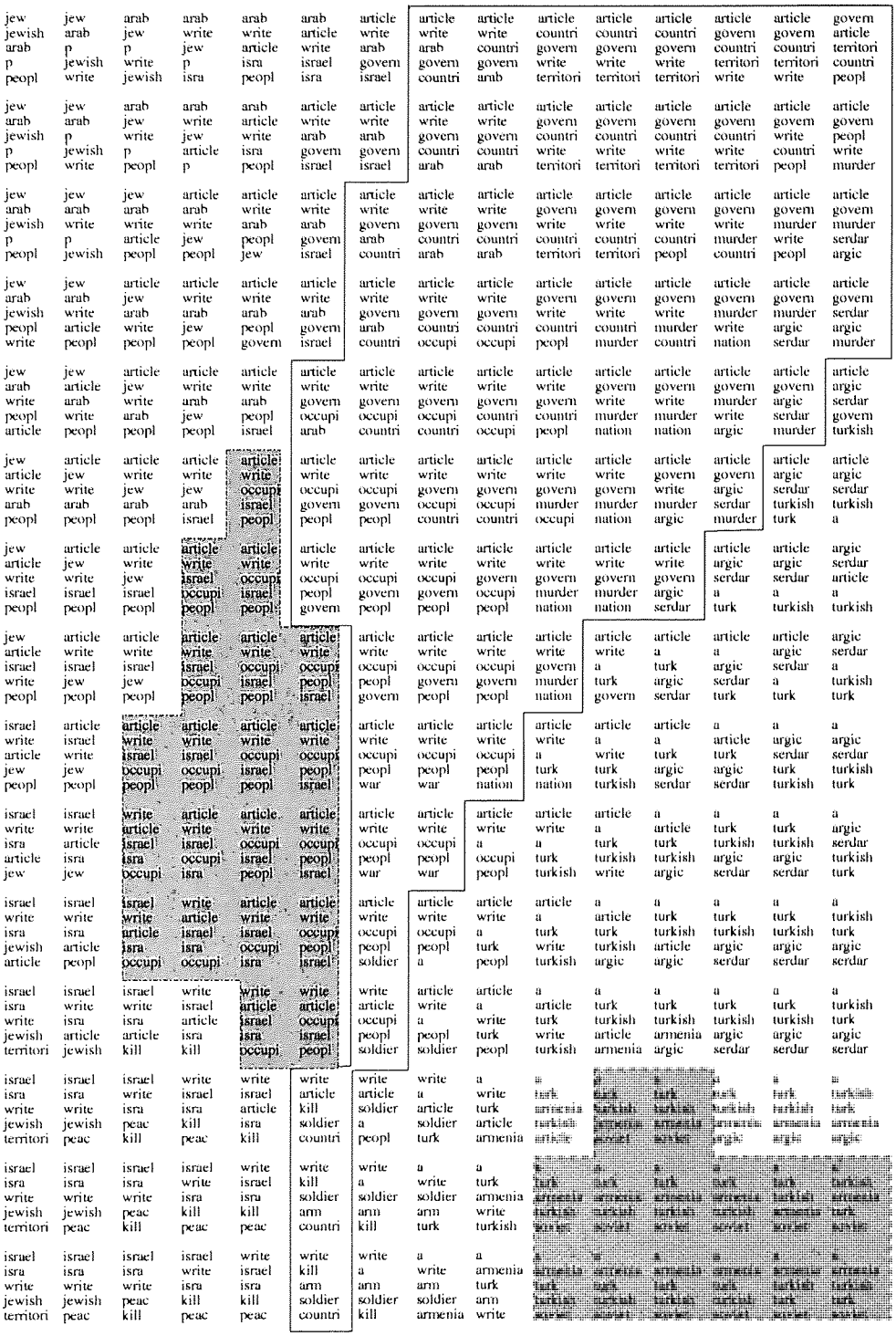
In this experiment, we randomly chose 60 percent data points from each class in the image segmentation dataset (see section 1.5.3) for training, and the remainder for testing. We trained a four-level hierarchy of LTMs with Gaussian noise model (GTMs) on the training set. Figures 4.14, 4.15 and 4.16 show the corresponding projection, magnification and curvature plots, respectively.

Looking at Figures 4.15 and 4.16, we can see that regions of high stretch and curvature correspond to those involving projections of the "outliers", which are mainly from class "Grass+Foliage", appearing in the bottom-left of the top level of Figure 4.14. This is clarified by the child-modulated ancestor plot technique, illustrated in Figure 4.17, where the position of data points captured by the first level-four GTM is highlighted in its ancestor plots.

For comparison, we also visualised the same dataset using PhiVis[2], which is a locally linear hierarchical visualisation algorithm developed by Bishop and Tipping (1998). Rather than showing a leaf model once in the whole hierarchy as in our system implement, PhiVis copies corresponding leaves models to lower levels and connects them using dotted lines. In addition, it displays the labeled orthogonal projections of the child visualisation planes onto their parent visualisation plot.

A hierarchical visualisation down to level five of the image training set obtained by PhiVis is presented in Figure 4.18. In the hierarchical GTMs, we can obtain a reasonable separation of those classes in the top level. Data points in the top level of Figure 4.14 are much more separately represented, whereas these segments merged when using PhiVis. Furthermore, we computed the negative log-likelihoods to test dataset for the two hierarchical models. For the hierarchy GTMs, its value is 0.085, while for Phivis, the value is 5.625. It indicated that the structure of the image segmentation data is not well captured by a simple linear transformation.

[2]A Matlab code for PhiVis is accessed at `http://www.ncrg.aston.ac.uk/PhiVis`.



Note: p denotes palestini; a says armenian.

Figure 4.12: The most probable words formed in each of the 15 by 15 latent grid points by the binomial latent trait model.

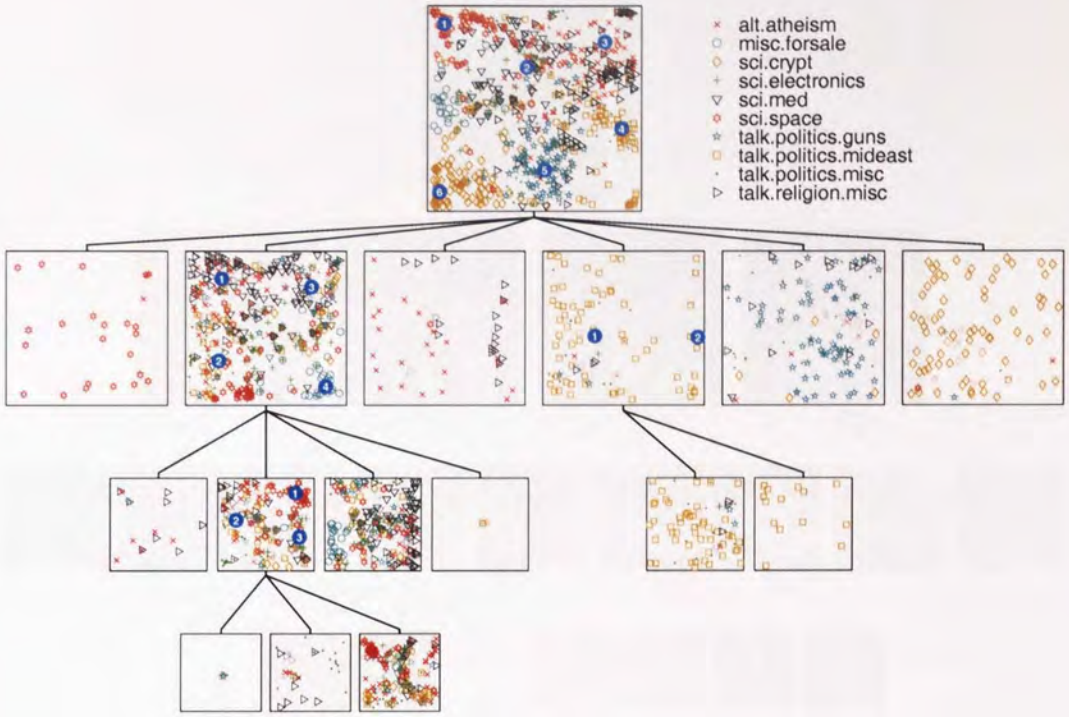


Figure 4.13: A visualisation plot of in the hierarchy of LTMs fitted to the document data without a EM training in the initialisation.

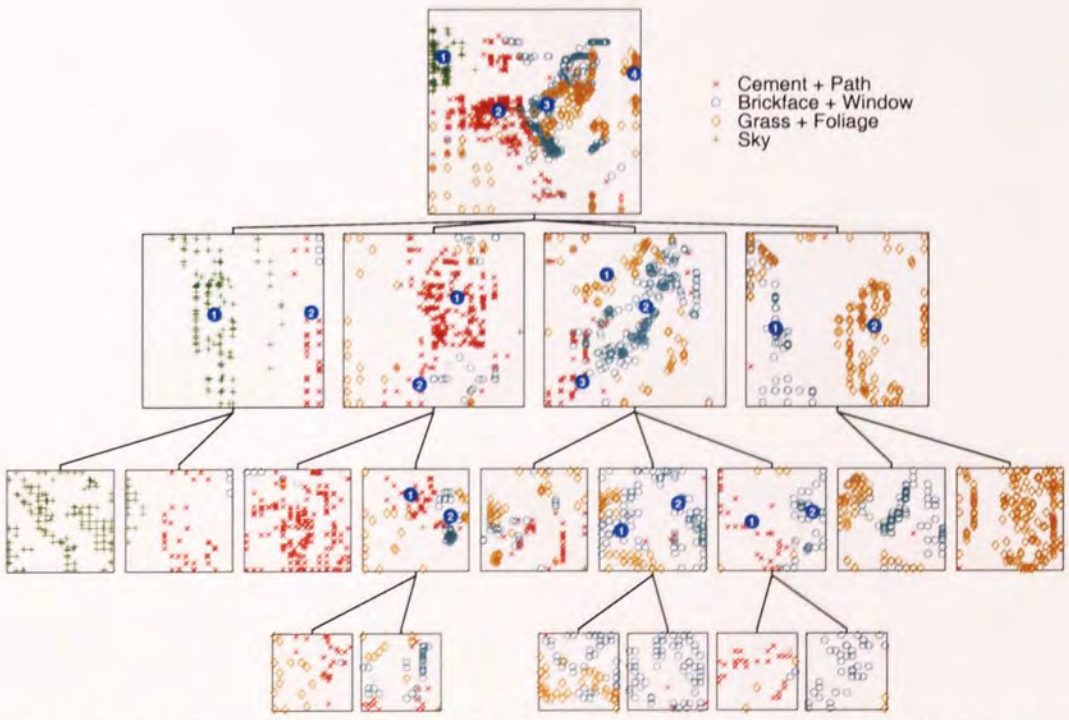


Figure 4.14: A visualisation plot of a hierarchy of GTMs fitted to the training set of the image data.

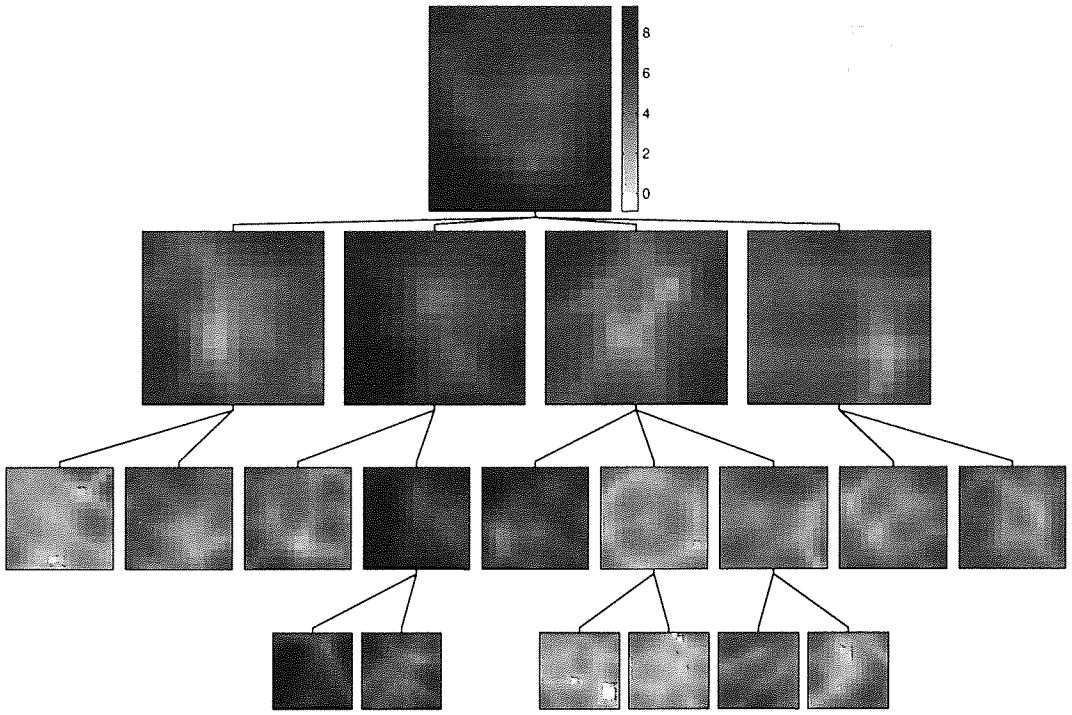


Figure 4.15: A magnification factor plot of a hierarchy of GTMs fitted to the training set of the image data.

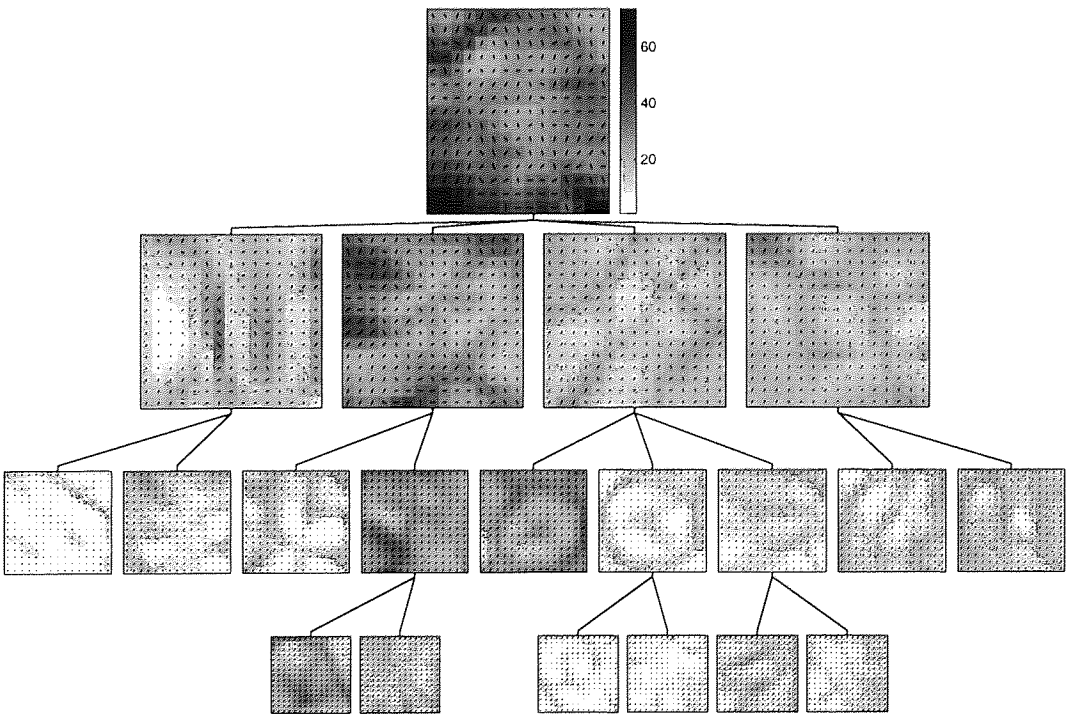


Figure 4.16: A curvature plot of a hierarchy of GTMs fitted to the training set of the image data.

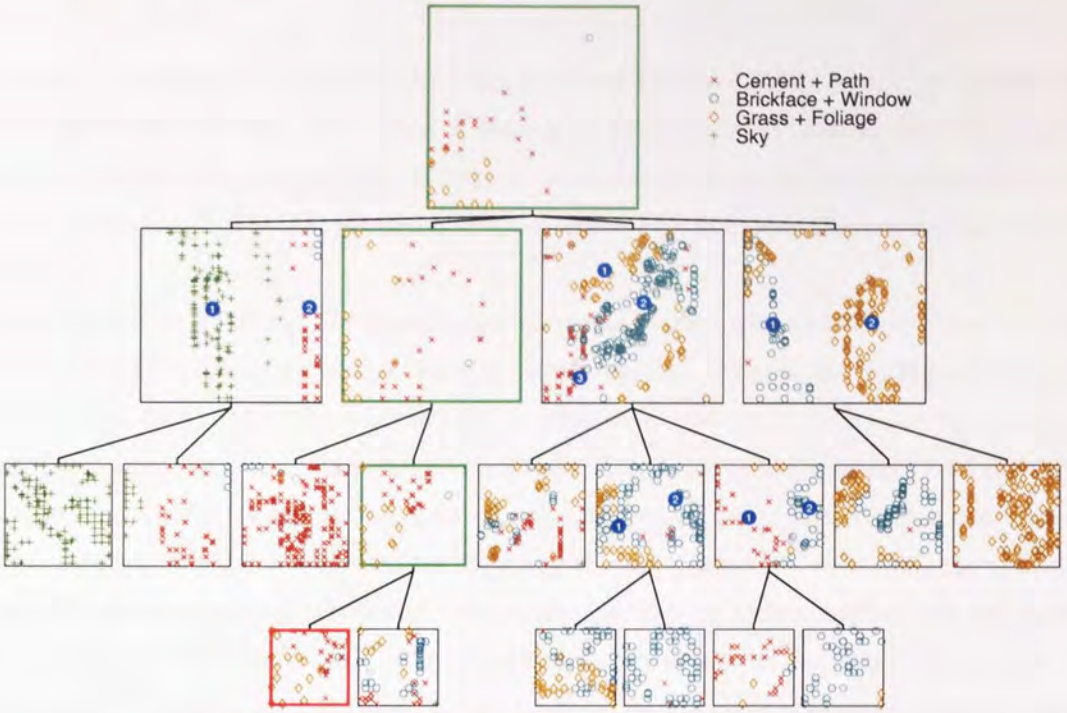


Figure 4.17: Hierarchical visualisation of the training set of the image data. The set of points captured by the first GTM at level four of the hierarchy is highlighted in the visualisation plots of all its ancestors.

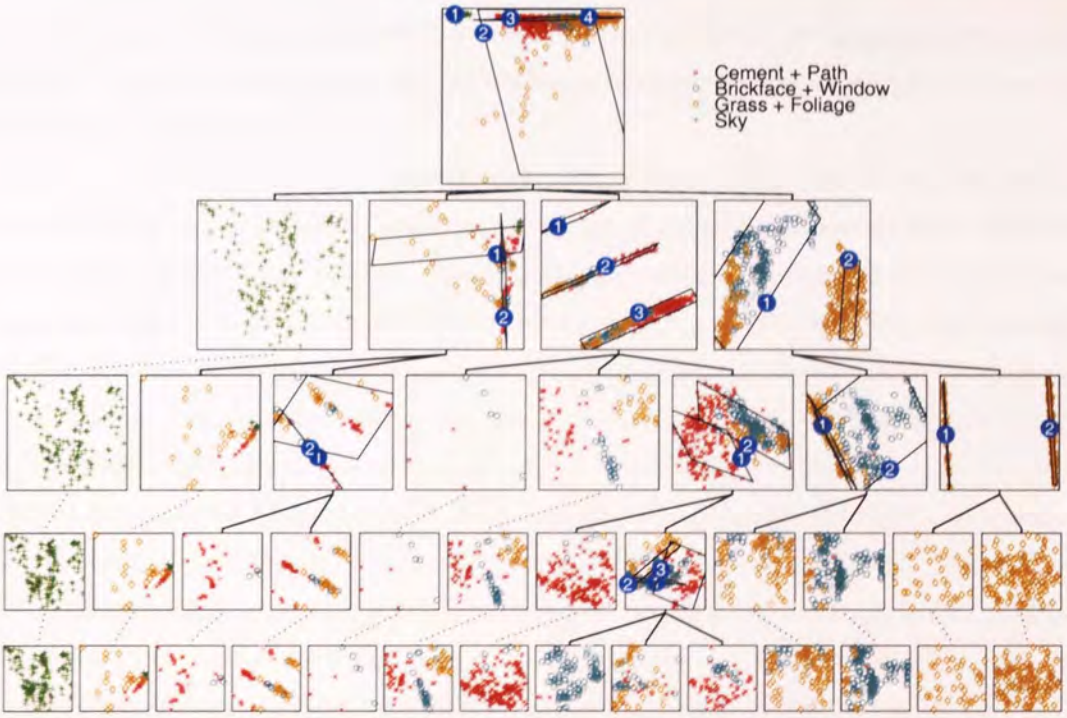


Figure 4.18: Phivis to the image data with a locally linear hierarchical visualisation algorithm. It copies corresponding leaves models to lower levels and connects them using dotted lines. In addition, it displays the orthogonal projections of the child visualisation planes onto the parent visualisation plot.

4.5.3 HTS data

In the last experiment we visualised the HTS data (see section 1.5.1). Again, we applied LTMs with Gaussian distributions. We trained a three-level hierarchy of GTMs on this data collection. The resulting projection, magnification factor and curvature plots are presented in Figures 4.19, 4.20 and 4.21, respectively. The models constructed by the GTM technique did not use any structural descriptors.

Researchers from Pfizer Central Research are interested in those data points which are close to each other but belonging to different classes. For example, in the first level-two plot in Figure 4.19, most of the points from the *0-active* class surround the *1-active* class, while the points from the *2-active* class are mostly grouped with those from the *1-active* class. Researchers further investigated these groups of compounds. The full chemical descriptors for the structure were obtained and a structure-based clustering tool was applied to determine the homology of compounds based on the most prominent ring system. The compounds were partitioned into groups that had significant overlap with the clustering observed in the GTM visualisations. The key distinction between the two techniques is that GTM utilizes mainly biological data and no structure information and the proprietary clustering tool uses only structural descriptors and no biological information.

Preliminary conclusions suggest that hierarchical visualisation using the GTM provides meaningful clustering of compound-related data based on biological information and a limited number of physio-chemical parameters. But more importantly the analysis suggests that there is potential for clustering compounds based on biological data which is meaningful in the light of grouping by purely structural parameters. This implies some similarity between how a Medical Chemist would view the compounds and the GTM visualisation.

One more interesting phenomenon arises from looking at Figure 4.22, given by the child-modulated ancestor plot technique. The points appearing at the top of the *root* plot, mainly from class *2-active* seem to belong to cluster 1 (i.e. the first level-two plot) with reference to Figure 4.19. In fact, however, they were captured by the second model (i.e. the second level-two models) and represented at the fifth level-three. This can be explained by Figure 4.21, the curvature plot. There exist high curvatures in the corresponding two areas. It indicates that the projection manifold was highly curved to fit those points belonging to cluster 2 in the data space.

Figure 4.23 shows the hierarchical visualisation down to level five of the HTS data obtained by PhiVis. Looking at the plot, we see that most of the active compounds were only separated at the third level (see the third and fourth sub-plots at this level and their parent at the second level), while this was achieved at the top level with HLTM (see Figure 4.19). In addition, we noticed that with a simple linear transformation, there were significant overlaps even at the fifth level.

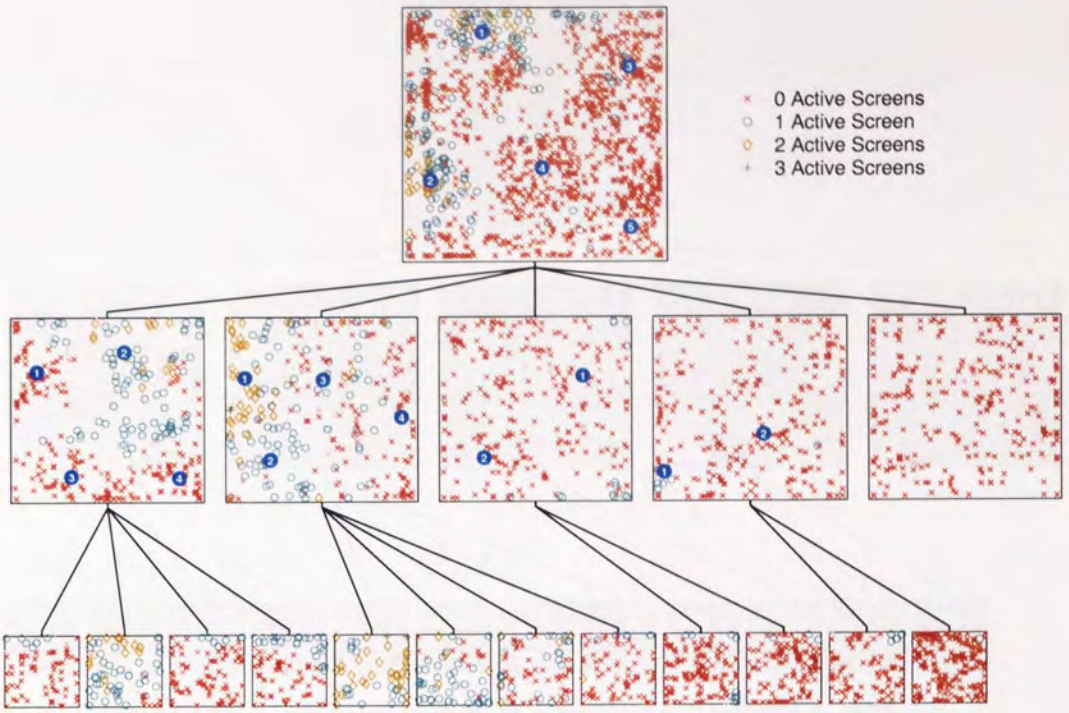


Figure 4.19: A visualisation plot of a hierarchy of LTMs fitted to HTS data.

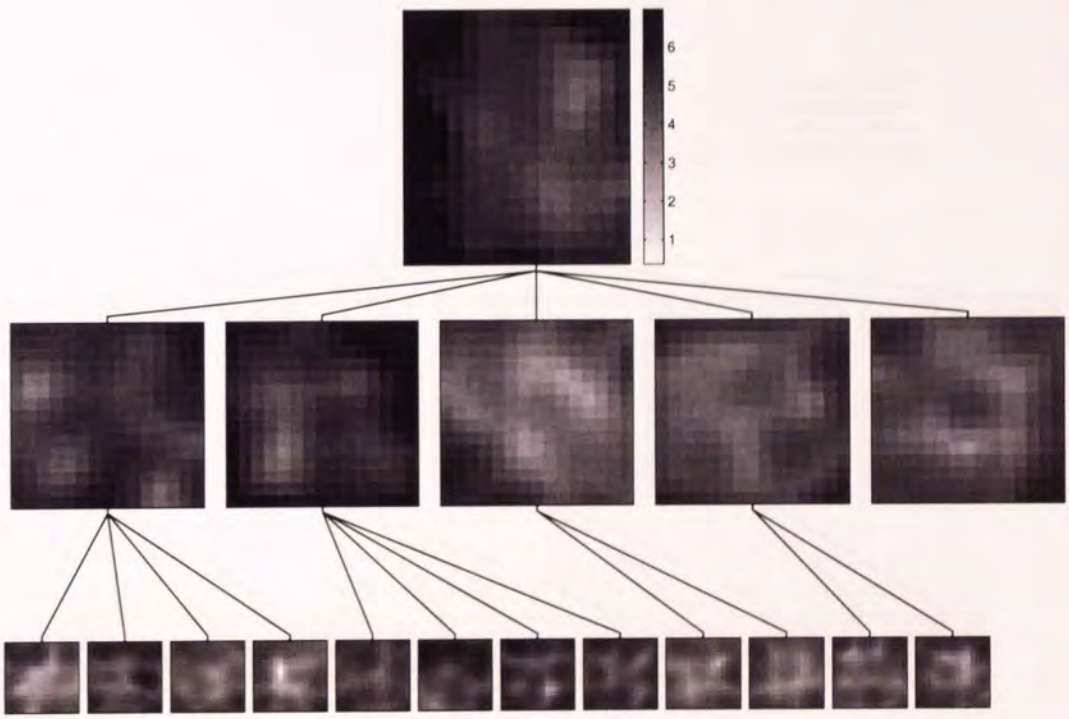


Figure 4.20: A magnification factor plot of a hierarchy of LTMs fitted to HTS data.

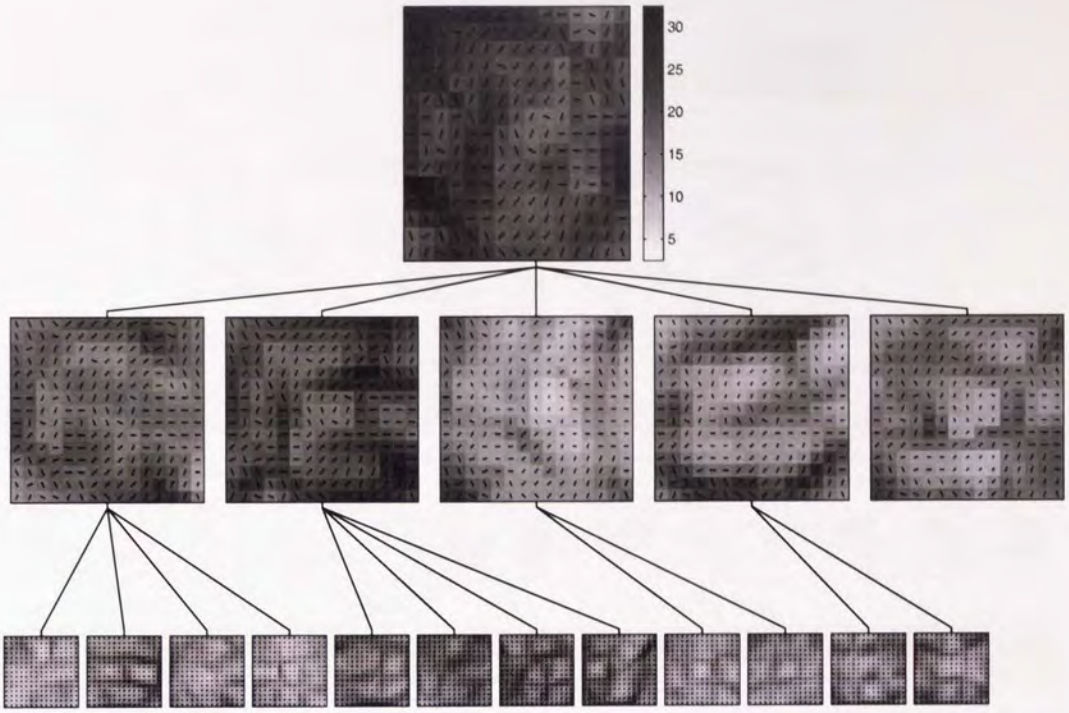


Figure 4.21: A curvature plot of a hierarchy of LTMs fitted to HTS data.

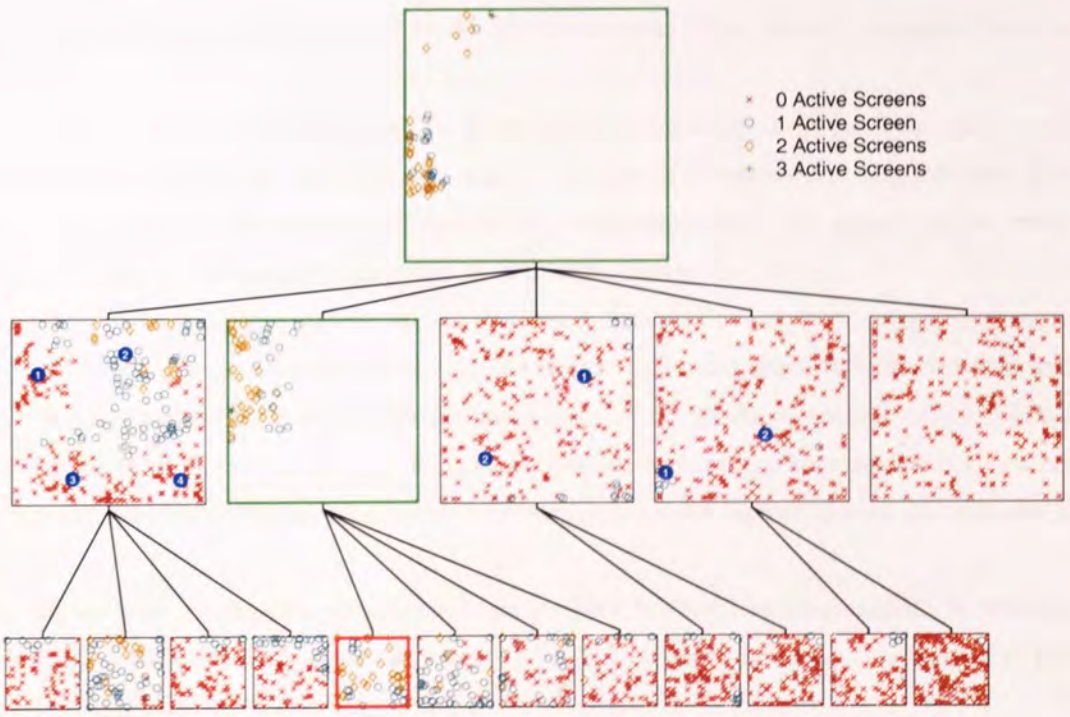


Figure 4.22: Hierarchical visualisation of the HTS data. The set of points captured by the fifth GTM at level three of the hierarchy is highlighted in the visualisation plots of all its ancestors.

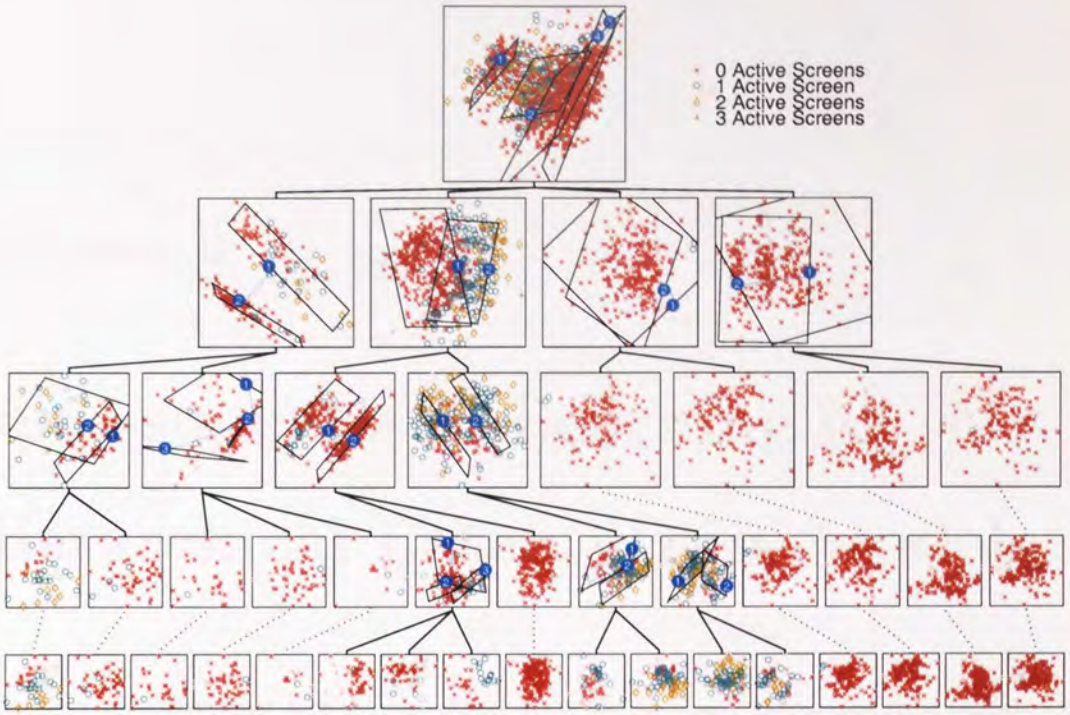


Figure 4.23: PhiVis to HTS data with a locally linear hierarchical visualisation algorithm.

4.6 Discussion

In this chapter, we have formally written out the full derivation for the hierarchical latent trait model. We have improved the initialisation of sub-models for discrete data, since the standard method does not work.

We have presented a general system for hierarchically visualising large datasets which may be of either continuous or discrete type. We also showed how use of curvature and magnification plots can help the user interpret the results and specify sub-model locations. The novelty of our work is in applying this sort of visualisation approach to HTS data for the first time.

Since our aim was to find structures over the image dataset, but not to do image compression, we used the whole properties described in 1.5.3. We believe that more distinct structures can be found when one just consider those properties on image colours. In addition, because of the limitation of the document data which are in a smaller property space with binary representations, in the result of document dataset, we can see that some documents cannot be separated even down to the deeper levels.

So far, we have developed a generalised framework for hierarchical visualisation. In this case, the user is allowed to select the regions of interest to further observe the data. This method is powerful when the clusters are separated clearly in the 2-D latent space.

Chapter 5

Semi-supervised Learning of Hierarchical Latent Trait Models

An interactive hierarchical latent trait model (HLTM) has been developed to visualise complex datasets. In this chapter, we extend the HLTM visualisation system by giving the user a choice of initialising the child plots of the current plot in either *interactive* or *automatic* mode. In the interactive mode the user selects "regions of interest" as in Chapter 4, whereas in the automatic mode an unsupervised minimum message length (MML)-driven construction of a mixture of LTMs is used.

The unsupervised construction is particularly useful when high-level plots are covered with dense clusters of strongly overlapping data projections, making it difficult to use the interactive mode. Such a situation often arises when visualising large datasets. We illustrate our approach on a synthetic example and apply our system to four more complex datasets. This MML algorithm can be combined seamlessly into an EM algorithm.

5.1 Introduction

So far, we have developed a general framework for a visualisation hierarchy. The user selects the "regions of interest" to refine the visualisation model. This method is powerful when the clusters are separated clearly in the 2-D latent (visualisation) space. On the other hand, when facing a "messy" plot like that in Figure 5.1, where thousands of data points are shown, the user may be unable to determine where sub-models should best be placed. In order to resolve this problem, we extend our current algorithm by providing an automated technique for deciding the number of sub-models and initialising their location.

The selection of the number of components in a mixture model is an important but very difficult problem. With too many components, mixture models may overfit the data, while a mixture with few components may not be flexible enough to approximate the true density distribution.

Most techniques used for deciding model complexity select a "best" model from a range of candi-

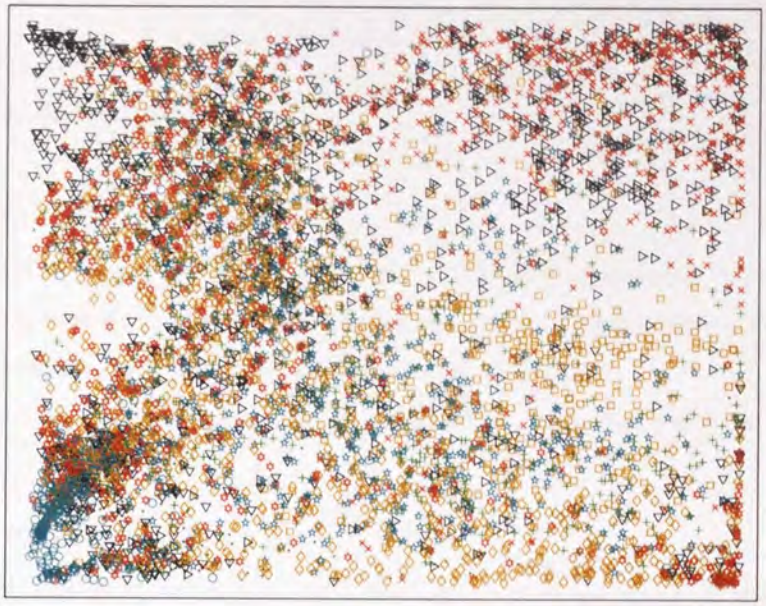


Figure 5.1: Visualisation of a large document dataset with an LTM.

dates, which have usually been obtained by the EM algorithm. In fact, these approaches separate the problem of assessing the number A of components from the fitting of the mixture model and estimating the parameters of A components. However, a more direct approach is to combine the determination of the number of components with parameter estimation, as discussed in (Figueiredo and Jain, 2002) based on a minimum message length criterion (Wallace and Dowe, 1999).

In the next section, we review some previous techniques used for assessing the number of mixture components. We explain how minimum message length (MML) can be employed for mixture models in section 5.3. Section 5.4 shows experimental results obtained with the semi-supervised learning of visualisation hierarchies. Finally, section 5.6 discusses the key conclusions of this chapter.

5.2 Previous work on assessing the number of components

Several approaches have been suggested for estimating the number of components. These can be classified into two types, deterministic methods and stochastic methods. However, stochastic techniques based on Markov chain Monte Carlo (MCMC) are still very computationally demanding (Figueiredo and Jain, 2002), and so we shall pay attention to deterministic methods in this section.

Deterministic methods use a two step procedure to decide the number of components. The first step is to obtain a set of candidate models (usually by the EM algorithm), denoted by $\{\mathcal{V}_a\}_{a_{min},\ldots,a_{max}}$. The second step involves computing the associated criterion and selecting $\hat{\mathcal{V}}$ which minimises (or maximises) the criterion. Note that in each candidate mixture model $\mathcal{V}_a$, we use $\boldsymbol{\theta}$ to denote the parameters of the whole mixture $\mathcal{V}_a$.

The number of components is then chosen according to

$$\hat{a} = \underset{a}{\operatorname{argmin}} \{E(\boldsymbol{\theta}, a), a = a_{min}, \ldots, a_{max}\}, \tag{5.1}$$

where E is some model selection criterion.

5.2.1 Selection criteria

Past work has used a wide range of solution criteria: Akaike's information criterion (AIC), informational complexity criterion (ICOMP), minimum description length (MDL), Schwarz's Bayesian information criterion (BIC) and minimum message length (MML).

We assume we have a dataset ζ including N independent observations of a random variable with probability density function $p(\mathbf{t})$.

- Akaike's Information Criterion (AIC)

  Akaike (1973 and 1974) developed a decision-making strategy based on the Kullback-Leibler information measure $\mathcal{D}_{KL}$ given by

  $$\mathcal{D}_{KL} = \int p(\mathbf{t}) \log p(\mathbf{t})\, d\mathbf{t} - \int p(\mathbf{t}) \log p(\mathbf{t}|\boldsymbol{\theta})\, d\mathbf{t}, \tag{5.2}$$

  where $p(\mathbf{t}|\boldsymbol{\theta})$ is our model of the density function, and $p(\mathbf{t})$ the (unknown) true density. Akaike suggested that the model be adopted giving the minimum of the expected $\mathcal{D}_{KL}$ from the true model. Note that the first term of the right hand side in equation (5.2) does not depend on the model. So we are interested in the expectation of the second term, which can be expressed as

  $$E = \langle \int p(\mathbf{t}) \log p(\mathbf{t}|\boldsymbol{\theta})\, d\mathbf{t} \rangle. \tag{5.3}$$

  In a model selection problem, Akaike framework proceeds by selecting a model having largest E. The problem is to find a consistent estimator of E. One of the most important characteristics of term $\int p(\mathbf{t}) \log p(\mathbf{t}|\boldsymbol{\theta})\, d\mathbf{t}$ is that its natural estimate, the average log-likelihood given by $\frac{1}{N} \log p(\zeta|\boldsymbol{\theta})$, can be obtained without the knowledge of $p(\mathbf{t})$ (Akaike, 1974). However, it can be shown that this estimator is biased. An information criterion for model selection can be based on the bias-connected log-likelihood given by

  $$\log p(\zeta|\boldsymbol{\theta}) - b, \tag{5.4}$$

  where b is a biased term.

  Akaike showed that term b is asymptotically equal to c, which is equal to the total number of parameters in the model. The information criteria are usually expressed in terms of twice of negative of equation (5.4). Thus Akaike information criterion, estimator of Kullback-Leibler information, can be written as

  $$\mathrm{AIC}(\mathcal{V}_a) = -2 \log p(\zeta|\boldsymbol{\theta}) + 2c. \tag{5.5}$$

The $\mathcal{V}_a$ having a componments is selected with the minimum value of AIC($\mathcal{V}_a$). There are a number of extensions of the AIC. One of them is proposed by Bozdogan (1983):

$$\text{AIC}(\mathcal{V}_a) = \frac{-2(N-1-q-\frac{A}{2})\log p(\zeta|\boldsymbol{\theta})}{N} + 3c, \tag{5.6}$$

where A is the largest number of components considered, c is the number of estimated parameters, and q is the number of parameters specifying each component of the mixture.

It has been observed that in the mixture context, AIC tends to overestimate the correct number of components (Celeux and Soromenho, 1996).

- Informational Complexity Criterion (ICOMP)

  ICOMP criterion is described in (Bozdogan, 1990; Bozdogan, 1993), which is aimed at improving on the performance of AIC. It has the form

  $$\text{ICOMP}(\mathcal{V}_a) = -2\log p(\zeta|\boldsymbol{\theta}) + C(\mathbf{I}(\boldsymbol{\theta})^{-1}), \tag{5.7}$$

  where

  $$C(\mathbf{I}(\boldsymbol{\theta})^{-1}) = c\log\left(\frac{trace(\mathbf{I}(\boldsymbol{\theta})^{-1})}{c}\right) - \log\left(|\mathbf{I}(\boldsymbol{\theta})^{-1}|\right), \tag{5.8}$$

  and $\mathbf{I}(\boldsymbol{\theta})$ is the expected Fisher Information matrix given by

  $$\mathbf{I}(\boldsymbol{\theta}) = \langle -\frac{\partial^2 \log p(\zeta|\boldsymbol{\theta})}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^T}\rangle, \tag{5.9}$$

  and c is the number of estimated parameters. We select the $\mathcal{V}_a$ with the minimum value of ICOMP($\mathcal{V}_a$). Celeux and Soromenho (1996) found that when the component covariance matrices were very different, ICOMP tends to overestimate the number of components in a mixture model.

- Schwarz's Bayesian Information Criterion (BIC)

  Rissanen (1986) derived his minimum description length (MDL) criterion based on coding theory (Cover and Thomas, 1991) for model selection. Schwarz (1978) independently developed the same criterion called Bayesian information criterion (BIC). The goal of the Bayesian analysis is to observe the data, compute the posterior probability of each model, and choose one having the highest posterior probability.

  For each mixture model with parameters $\boldsymbol{\theta}$, there is a prior density denoted by $p(\boldsymbol{\theta})$. The integrated likelihood $p(\zeta)$ is given by

  $$\begin{aligned} p(\zeta) &= \int p(\boldsymbol{\theta},\zeta)\, d\boldsymbol{\theta} \\ &= \int \exp\{\log p(\boldsymbol{\theta},\zeta)\}\, d\boldsymbol{\theta}, \end{aligned} \tag{5.10}$$

  where $p(\boldsymbol{\theta},\zeta) = p(\boldsymbol{\theta})p(\zeta|\boldsymbol{\theta})$.

  To approximate the integral (5.10), Laplace's approach is used. First we let $\hat{\boldsymbol{\theta}}$ to denote the posterior mode, satisfying

  $$\frac{\partial \log p(\hat{\boldsymbol{\theta}},\zeta)}{\partial\boldsymbol{\theta}} = 0. \tag{5.11}$$

Employing a second order Taylor series expansion of $\log p(\boldsymbol{\theta}, \zeta)$ around the maximum a posterior (MAP) estimate $\hat{\boldsymbol{\theta}}$, we obtain

$$\log p(\boldsymbol{\theta}, \zeta) \approx \log p(\hat{\boldsymbol{\theta}}, \zeta) - \frac{1}{2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T H(\hat{\boldsymbol{\theta}})(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}), \tag{5.12}$$

where $H(\hat{\boldsymbol{\theta}})$ denotes the negative Hessian matrix of $\log p(\boldsymbol{\theta}, \zeta)$ evaluated at $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$,

$$H(\hat{\boldsymbol{\theta}}) = \left[-\frac{\partial^2}{\partial \boldsymbol{\theta}_i \partial \boldsymbol{\theta}_j} \log p(\boldsymbol{\theta}, \zeta) \right]_{|_{\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}}}. \tag{5.13}$$

Substituting this approximation for $\log p(\boldsymbol{\theta}, \zeta)$ into equation (5.10), yields

$$\begin{aligned} p(\zeta) &= \exp\{\log p(\hat{\boldsymbol{\theta}}, \zeta)\} \int \exp\left\{ \frac{1}{2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T H(\hat{\boldsymbol{\theta}})(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) \right\} d\boldsymbol{\theta} \\ &= p(\hat{\boldsymbol{\theta}}, \zeta)(2\pi)^{\frac{c}{2}} |H(\hat{\boldsymbol{\theta}})|^{-\frac{1}{2}}. \end{aligned} \tag{5.14}$$

In terms of logarithms, the integrated log-likelihood is approximated as

$$\log p(\zeta) = \log p(\hat{\boldsymbol{\theta}}) + \log p(\zeta|\hat{\boldsymbol{\theta}}) - \frac{1}{2} \log |H(\hat{\boldsymbol{\theta}})| + \frac{c}{2} \log(2\pi). \tag{5.15}$$

An important variant on equation (5.15) can be obtained by assuming that the prior is sufficiently flat. In this case, by using Laplace's approach, we have

$$\log p(\zeta) = \log p(\tilde{\boldsymbol{\theta}}) + \log p(\zeta|\tilde{\boldsymbol{\theta}}) - \frac{1}{2} |\mathcal{I}(\tilde{\boldsymbol{\theta}}, \zeta)| + \frac{c}{2} \log(2\pi), \tag{5.16}$$

where $\tilde{\boldsymbol{\theta}}$ is a maximum likelihood estimate and $\mathcal{I}(\tilde{\boldsymbol{\theta}})$ is the observed information matrix, given by $-\frac{\partial^2 \log p(\zeta|\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T}$. In equation (5.16), when $N \to \infty$, the terms which are a function of N begin to dominate the others, and we obtain the BIC formula, given by

$$\text{BIC}(\mathcal{V}_a) = -2 \log p(\zeta|\hat{\boldsymbol{\theta}}) + c \log N. \tag{5.17}$$

We select the model $\mathcal{V}_a$ with the minimum value of $\text{BIC}(\mathcal{V}_a)$.

- The Wallace-Freeman minimum message length (MML)

  Minimum message length (MML) (Wallace and Dowe, 1999) strategies select, among the models inferred from ζ, the one which minimises length of the message transmitting ζ. The message consists of two parts – one specifying the model parameters, the other specifying the data given the model:

  $$\text{Length}(\boldsymbol{\theta}, \zeta) = \text{Length}(\boldsymbol{\theta}) + \text{Length}(\zeta|\boldsymbol{\theta}). \tag{5.18}$$

  By Shannon theory (Cover and Thomas, 1991), $\text{Length}(\boldsymbol{\theta})$ is no less than $\lceil -\log P(\boldsymbol{\theta}) \rceil$ (based on a prior over the model space), and $\text{Length}(\zeta|\boldsymbol{\theta})$ no less than $\lceil -\log(p(\zeta|\boldsymbol{\theta})) \rceil$. $\lceil l \rceil$ denotes the smallest integer no less than l. We will neglect rounding to integer values.

  Here we introduce a particular form of the MML approach, offered by Wallace and Freeman (1987), which is given by

  $$\jmath(\boldsymbol{\theta}, \zeta) = -\log p(\boldsymbol{\theta}) - \log p(\zeta|\boldsymbol{\theta}) + \frac{1}{2} \log |\mathbf{I}(\boldsymbol{\theta})| + \frac{c}{2}(1 + \log \kappa_c), \tag{5.19}$$

where $\mathbf{I}(\boldsymbol{\theta})$ is the expected Fisher information matrix, having the form

$$\begin{aligned} \mathbf{I}(\boldsymbol{\theta}) &= -\langle \frac{\partial^2 \log p(\zeta|\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \rangle \\ &= -\int p(\zeta|\boldsymbol{\theta}) \frac{\partial^2 \log p(\zeta|\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \, d\zeta, \end{aligned} \tag{5.20}$$

and κ_c is a c-dimensional optimal quantizing lattice constant. We select the model with the minimum values of $\jmath(\boldsymbol{\theta}, \zeta)$. A detailed review of the Wallace-Freeman MML can be found in Appendix D.

A more detailed review can be found in (McLachlan and Peel, 2000, Chapter 6), where other approaches, such as nonparametric methods including a number of graphical tools like histograms and normal scores plots (Cassie, 1954; Harding, 1948), and methods of moments used to test for the number of components in cluster analysis (Dacunha-Castelle and Gassiat, 1997; Vlassis and Likas, 1999), are described.

5.2.2 Penalized log-likelihood interpretation

The model selection procedures described above can be expressed as penalized log-likelihoods (Green, 1998):

$$E_{pen} = \log p(\zeta|\boldsymbol{\theta}) - \wp(\boldsymbol{\theta}, \zeta). \tag{5.21}$$

The penalties associated with Schwarz's approach (5.16) and the MML (5.19) are given respectively by

$$\wp_s = -\log p(\hat{\boldsymbol{\theta}}) + \frac{1}{2} \log |H(\hat{\boldsymbol{\theta}})| - \frac{c}{2} \log(2\pi), \tag{5.22}$$

$$\wp_w = -\log p(\boldsymbol{\theta}) + \frac{1}{2} \log(\mathbf{I}(\boldsymbol{\theta})) + \frac{c}{2}(1 + \log \kappa_c). \tag{5.23}$$

For $\wp_s$, $\hat{\boldsymbol{\theta}}$ is the MAP estimate in order to make the Laplace approximation, whereas for $\wp_w$, $\boldsymbol{\theta}$ is the one which minimises $\wp_w$.

5.2.3 Summary of previous work

In (Oliver *et al.*, 1996), authors gave an empirical comparison of criteria. They concluded that the MML criterion performs better than some other criteria, such as AIC, BIC and ICOMP. Particularly, they found that the MML criterion is more conservative than AIC, BIC and ICOMP criteria. That means, when the criteria cannot test the correct number (the actual number is known as they generated the data from some predefined distributions) of components, the MML criterion is more likely to predict the number of components less than the true values, while AIC, BIC and ICOMP criteria often gave more components than the real values. In fact, this is a drawback of these methods since the assessed mixture may include some models having zero mixture probabilities. For example, a 3-component mixture including one whose mixing coefficients approaches zero may not be distinguished from a 2-component mixture without that "dying" model.

A direct approach is to find the "best" overall model in the whole set of available models $\{\mathcal{V}_a\}_{a_{min},\ldots,a_{max}}$ directly, rather than selecting one among a set of candidate models. This, in fact, is the principle of the MML principle (Wallace and Dowe, 1999). Previous uses of the MML as a model selection criterion for mixtures do not strictly adhere to this perspective.

5.3 MML formulation for unsupervised learning of mixture models

5.3.1 MML formulation for unsupervised learning of mixture models

Recently, Figueiredo and Jain (2002) extended the MML framework to unsupervised learning of mixture models; the algorithm is able to select the "appropriate" number of components while the parameters of each model are estimated in the usual way. The novelty of their proposed approach is that parameter estimation and model selection are integrated in a single algorithm, rather than using a model selection criterion on a set of pre-estimated candidate models. Their approach can be applied to any type of parametric mixture model for which it is possible to write an EM algorithm. In this section, we briefly reformulate in our notation their key results.

The particular form of the MML criterion adopted in (Figueiredo and Jain, 2002) is of the form $\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \jmath(\boldsymbol{\theta}, \zeta)$, where

$$\jmath(\boldsymbol{\theta}, \zeta) = -\log p(\boldsymbol{\theta}) - \log p(\zeta|\boldsymbol{\theta}) + \frac{1}{2}\log|\mathbf{I}(\boldsymbol{\theta})| + \frac{c}{2}\left(1 + \log\frac{1}{12}\right), \tag{5.24}$$

where $\mathbf{I}(\boldsymbol{\theta})$ is the expected Fisher information matrix, $|\mathbf{I}(\boldsymbol{\theta})|$ is the determinant of $\mathbf{I}(\boldsymbol{\theta})$, and c is the dimension of $\boldsymbol{\theta}$. Comparing with equation (5.19), they approximated κ_c by $\frac{1}{12}$, which arises from hyper-cubic quantization regions when $c = 1$. Actually, κ_c does not vary much and approaches an asymptotic value, $(2\pi e)^{-1} \approx 0.05855$ (Conway and Sloane, 1993).

In (Figueiredo and Jain, 2002), $\mathbf{I}(\boldsymbol{\theta})$ was replaced by the complete-data[1] Fisher information matrix $\mathbf{I}_c(\boldsymbol{\theta})$ since, in general, $\mathbf{I}(\boldsymbol{\theta})$ cannot be obtained analytically. $\mathbf{I}_c(\boldsymbol{\theta})$ upper-bounds $\mathbf{I}(\boldsymbol{\theta})$ (Titterington *et al.*, 1985) and has a block-diagonal structure

$$\mathbf{I}_c(\boldsymbol{\theta}) = \mathrm{N}\ \text{block-diag}\{P(1)\mathbf{I}^{(1)}(\boldsymbol{\theta}_1), \ldots, P(A)\mathbf{I}^{(1)}(\boldsymbol{\theta}_A), \mathbf{F}\},$$

where $P(a), a = 1, \ldots, A$ are the mixing coefficients; $\mathbf{I}^{(1)}(\boldsymbol{\theta}_a)$, is the Fisher information matrix for a single observation produced by the a-th mixture component, and $\mathbf{F}$ is the Fisher matrix of a multinomial distribution over mixing coefficients, whose determinant is $|\mathbf{F}| = (P(1)P(2)\cdots P(A))^{-1}$ (Titterington *et al.*, 1985). Then we have

$$\begin{aligned}\log|\mathbf{I}_c(\boldsymbol{\theta})| &= \log|N\mathbf{I}_{dc}| + \sum_{a=1}^{A}\log|P(a)\mathbf{I}_{dQ}| + \sum_{a=1}^{A}\log|\mathbf{I}^{(1)}(\boldsymbol{\theta}_a)| - \sum_{a=1}^{A}\log P(a) \\ &= c\log N + \sum_{a=1}^{A}\log|\mathbf{I}^{(1)}(\boldsymbol{\theta}_a)| + (Q-1)\sum_{a=1}^{A}\log P(a),\end{aligned} \tag{5.25}$$

[1] Training set ζ extended with the set of assignment variables for mixture components.

where $\mathbf{I}_{dc}$ and $\mathbf{I}_{dQ}$ are the $c \times c$ and $Q \times Q$ identity matrices respectively, and Q is the number of free parameters in each individual mixture component. We assume that the priors are independent, which means

$$p(\boldsymbol{\theta}) = p(P(1), \ldots, P(A)) \prod_{a=1}^{A} p(\boldsymbol{\theta}_a). \tag{5.26}$$

A non-informative Jeffreys' prior (Bernardo and Smith, 1994) is imposed on both the vector of mixing coefficients $\{P(a)\}$ and the parameters $\boldsymbol{\theta}_a$ of individual mixture components:

$$p(\boldsymbol{\theta}_a) \propto \sqrt{|\mathbf{I}^{(1)}(\boldsymbol{\theta}_a)|} \tag{5.27}$$

$$p(P(1), \ldots, P(A)) \propto \sqrt{|\mathbf{F}|} = (P(1)P(2)\cdots P(A))^{-1/2}, \tag{5.28}$$

for

$$0 \leq P(1), \ldots, P(A) \leq 1 \qquad \text{and} \qquad \sum_{a=1}^{A} P(A) = 1. \tag{5.29}$$

For an A-component mixture, $c = QA + A$. Now the equation (5.24) becomes

$$\jmath(\boldsymbol{\theta}, \zeta) = \frac{Q}{2} \sum_{a=1}^{A} \log\left(\frac{NP(a)}{12}\right) + \frac{A}{2} \log \frac{N}{12} + \frac{A(Q+1)}{2} - \log p(\zeta|\boldsymbol{\theta}). \tag{5.30}$$

The objective function in equation (5.30) does not make sense if we allow any of the $P(a)$'s to be zero (Figueiredo and Jain, 2002). Since we want a short code for the data, to specify the mixture model, we only code the parameters of mixture components a with positive prior $P(a)$. The number of such components is denoted by A_+. We obtain:

$$\jmath(\boldsymbol{\theta}, \zeta) = \frac{Q}{2} \sum_{a:P(a)>0} \log\left(\frac{NP(a)}{12}\right) + \frac{A_+}{2} \log \frac{N}{12} + \frac{A_+(Q+1)}{2} - \log p(\zeta|\boldsymbol{\theta}). \tag{5.31}$$

Details concerning the derivation of (5.31) are given in Appendix E.1.

Minimisation of (5.31) with respect to $P(a)$ must consider the constraints (5.29). This can be achieved by introducing a Lagrange multiplier λ. With A_+ fixed, we obtain the following re-estimation formulas for the mixture coefficients in the $M-$step (Figueiredo and Jain, 2002) (the derivation is presented in Appendix E.2)

$$\hat{P}(a) = \frac{\max\left\{0,\ -\frac{Q}{2} + \sum_{n=1}^{N} P(a|\mathbf{t}_n)\right\}}{\sum_{a'=1}^{A_+} \max\left\{0,\ -\frac{Q}{2} + \sum_{n=1}^{N} P(a'|\mathbf{t}_n)\right\}}, \qquad a = 1, 2, \ldots, A_+, \tag{5.32}$$

where component responsibilities $P(a|\mathbf{t}_n)$ are determined by the $E-$step

$$P(a|\mathbf{t}_n) = \frac{P(a)p(\mathbf{t}_n|a)}{\sum_{a'=1}^{A} P(a')p(\mathbf{t}_n|a')}, \tag{5.33}$$

in which case, the mixing coefficients are given by

$$P(a) = \frac{\sum_{n=1}^{N} P(a|\mathbf{t}_n)}{\sum_{a'=1}^{A} \sum_{n=1}^{N} P(a'|\mathbf{t}_n)}; \tag{5.34}$$

and other parameters in the M-step:

$$\hat{\boldsymbol{\theta}}_a = \underset{\boldsymbol{\theta}_a}{\operatorname{argmin}} \jmath(\hat{\boldsymbol{\theta}}, \zeta). \tag{5.35}$$

Free parameters of the individual LTMs are fitted to the data ζ using the EM algorithm outlined in section 4.3 applied to mixtures of LTMs[2]. Note that LTMs corresponding to zero $\hat{P}(a)$ become irrelevant and so equation (5.32) effectively performs component annihilation (Figueiredo and Jain, 2002).

5.3.2 The algorithm for mixture latent trait models

Note that in this section we will just focus on the algorithm for mixture latent trait models.
Given the training data ζ, we use the MML approach to find the "appropriate" number of mixture component LTMs that "explain" ζ in a probabilistic manner. LTMs that are good probabilistic generating models of the data capture the data distribution well and hence yield "good" visualisation plots, which means the projection manifold follows closely the data distribution and so the visualisation plot is a "good" representation of the data distribution. To start the training process, we choose the maximum number of components A_{max} we are willing to consider. The A_{max} points are selected randomly from the training dataset in the data space, and correspond to those points $\boldsymbol{\Omega}(\mathbf{c})$ discussed in section 4.3.5. Then, we initialise the component LTMs using the method described in section 4.3.5.

However, if we directly use EM with the M-step in equations (5.32) and (5.35), it may happen that $\sum_{n=1}^{N} P(a|\mathbf{t}_n) < \frac{Q}{2} (a = 1, \cdots, A_{max})$, where A_{max} is too large. That leads priors to be zero and components without enough initial support. As in (Figueiredo and Jain, 2002), we adopt the component-wise EM (CEM) algorithm (see Section 2.2.2), i.e. rather than simultaneously updating all the LTMs, we first update the parameters $\boldsymbol{\theta}_1$ of the first LTM by equation (4.27), while the parameters of the remaining LTMs are fixed, then we recompute the model responsibilities $\{P(a|\mathbf{t}_n)\}_{a=1}^{A}$ by equation (5.33) and mixture coefficients $\hat{P}(a)$ for all components in the mixture. After this, we move to the second component, update $\boldsymbol{\theta}_2$ in the same way, and recompute $\{P(a|\mathbf{t}_n)\}_{a=1}^{A}$, etc., looping through all mixture components. If one of the component LTMs dies ($\hat{P}(a) = 0$), redistribution of its probability mass to the remaining components increases their chance of survival.

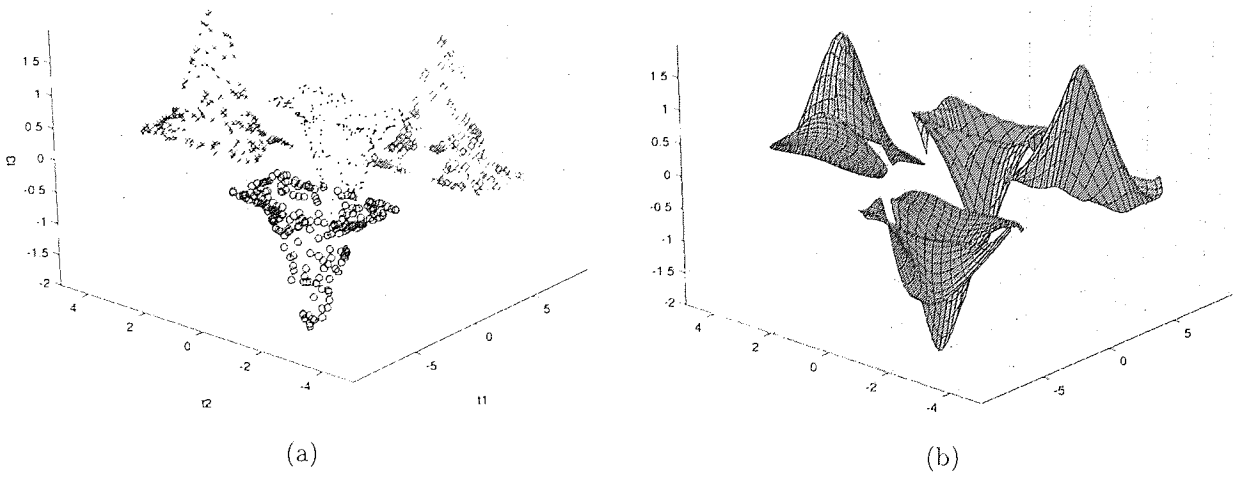
After convergence of CEM, we still have to check whether a shorter message length can be achieved by having a smaller number of mixture LTMs (down to $A_+ = 1$).[3] This is because equation (5.32) obtained with fixed A_+ does not consider the additional decrease in $\jmath(\boldsymbol{\theta}, \zeta)$ caused by the decrease in A_+. So we simply iteratively kill off the weakest LTM (with the smallest $\hat{P}(a)$) and re-run CEM until convergence. Finally, the winning mixture of LTMs is the one that leads to the shortest message length $\jmath(\boldsymbol{\theta}, \zeta)$ (see equation (5.31)). Since the LTM itself is a mixture model, it involves an EM algorithm (see section 2.6.1) in updating the parameters $\boldsymbol{\theta}_a$ of the ath LTM when going through each LTM within the CEM procedure for a mixture of LTMs. This case is different from the simple Gaussian mixture

[2] A mixture of LTMs can be considered as a two-level hierarchical LTM. Mixture components are children of the *root*.
[3] If we knew that the number of mixture components was no less than some number A_{min}, we would stop at $A_+ = A_{min}$ (Figueiredo and Jain, 2002).

model. When using CEM for Gaussian mixture models, the parameters of each Gaussian model are estimated by those one-step equations, which are equation (2.19) for updating the corresponding mixture coefficient, equation (2.20) for the mean and equation (2.21), (2.22) or (2.23) for covariance matrix.

Empirically, we observed that "strong" LTMs which survived for longer time periods tended to be over-trained. One does not encounter such problems when dealing with simple mixtures of Gaussians. However, the LTM is a constrained mixture in that the "centres", $\mathbf{\Omega}(\mathbf{x}_k)$ (equation (2.76)), cannot move separately. Therefore, we adopted the following technique: after a component LTM has been eliminated and before starting a new competition of the remaining LTMs for the data explained by it, we re-initialise the remaining LTMs so that they remain in their respective positions determined by the MML-based technique, but have a "fresh start" with less complicated projection manifolds. For each LTM we collect the data points for which that LTM has responsibility (equation (5.33)) higher than a threshold $\Delta = 0.80 \sim 0.85$. We then initialise and train individual LTMs for 1 epoch in the traditional way (Bishop *et al.*, 1998; Kabán and Girolami, 2001), each on the corresponding model-restricted set, as if they were not members of a mixture. After this re-initialisation step, the CEM algorithm is applied to the full mixture on the whole dataset. A detailed pseudo-code description of the algorithm is listed in Table 5.1.

To illustrate this algorithm, we did an experiment on a toy dataset of 800 points $\mathbf{t} = (t_1, t_2, t_3)^T$ lying on four two-dimensional manifolds ("humps") (see Figure 5.2 (a)). We associated the points in the four "humps" with four different classes, $\mathcal{C}_i$, $i = 1, 2, 3, 4$, having four different labels. After training ($A_{max} = 10$), a 6-component mixture was constructed. Projection manifolds of the 6 LTMs are shown in Figure 5.2 (b). Note that 6 child plots provide understandable subgroups of the data; and that the 6 projection manifolds closely approximate the four "humps" of the original generating manifold. The corresponding hierarchy of visualisation plots can be seen in Figure 5.3. Figure 5.4 reveals the



(a) (b)

Figure 5.2: (a) A two dimensional manifolds in data space; (b) Projection manifolds in data space of the second-level LTMs trained on the toy data.

Inputs: $A_{min}, A_{max}, \epsilon$, *niters*, initial parameters $\hat{\boldsymbol{\theta}}(0) = \{\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_{A_{max}}, \hat{P}(1), \ldots, \hat{P}(A_{max})\}$
Output: Mixture model in $\hat{\boldsymbol{\theta}}_{best}$

$j \leftarrow 0, \quad A_+ \leftarrow A_{max}, \quad \jmath_{min} \leftarrow +\infty$
$u_a^{(n)} \leftarrow p(\mathbf{t}_n|\boldsymbol{\theta}_a)$, for $a = 1, \ldots, A_{max}$, and $n = 1, \ldots, N$
while $A_+ > A_{min}$ **do**
  repeat
    $j \leftarrow j + 1$
    for $a = 1$ to A_+ **do**
      for *loops* $= 1$ to *niters* **do**
        E-step
          Compute posteriori: equation (2.72); rescaled by (5.33)
        M-step
          update the parameters of current LTM: equation (4.27)
      end for
      Compute equation (5.32)
      $\{\hat{P}(1), \ldots, \hat{P}(A_{max})\} \leftarrow \{\hat{P}(1), \ldots, \hat{P}(A_{max})\}(\sum_{a=1}^{A_{max}} P(a))^{-1}$
      if $\hat{P}(a) == 0$
        $A_+ = A_+ - 1$;
        initialise the remainder of components using data which has high responsibilities, while keeping the corresponding priors' values.
      end if
    end for
    $\hat{\boldsymbol{\theta}}(j) \leftarrow \{\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_{A_{max}}, \hat{P}(1), \ldots, \hat{P}(A_{max})\}$,
    $\jmath(\hat{\boldsymbol{\theta}}(j), \zeta) \leftarrow \frac{Q}{2} \sum_{a:P(a)>0} \log\left(\frac{N\hat{P}(a)}{12}\right) + \frac{A_+}{2} \log\frac{N}{12} + \frac{A_+(Q+1)}{2} - \sum_{n=1}^{N} \log \sum_{a=1}^{A_+} \hat{P}(a) u_a^{(n)}$.
  until $\jmath(\hat{\boldsymbol{\theta}}(j-1), \zeta) - \jmath(\hat{\boldsymbol{\theta}}(j), \zeta) < \epsilon\,|\jmath(\hat{\boldsymbol{\theta}}(j-1), \zeta)|$
  if $\jmath(\hat{\boldsymbol{\theta}}(j), \zeta) \leq \jmath_{min}$ **then**
    $\jmath_{min} \leftarrow \jmath(\hat{\boldsymbol{\theta}}(j), \zeta)$
    $\hat{\boldsymbol{\theta}}_{best} \leftarrow \hat{\boldsymbol{\theta}}(j)$
  end if
  $a^* \leftarrow \underset{a}{\text{argmin}}\{\hat{P}(a) > 0\}, \quad \hat{P}(a^*) \leftarrow 0, \quad A_+ \leftarrow A_+ - 1,$
  initialise the remainder of components using data which has high responsibilities, while keeping the corresponding priors' values.
end while

Table 5.1: The complete algorithm

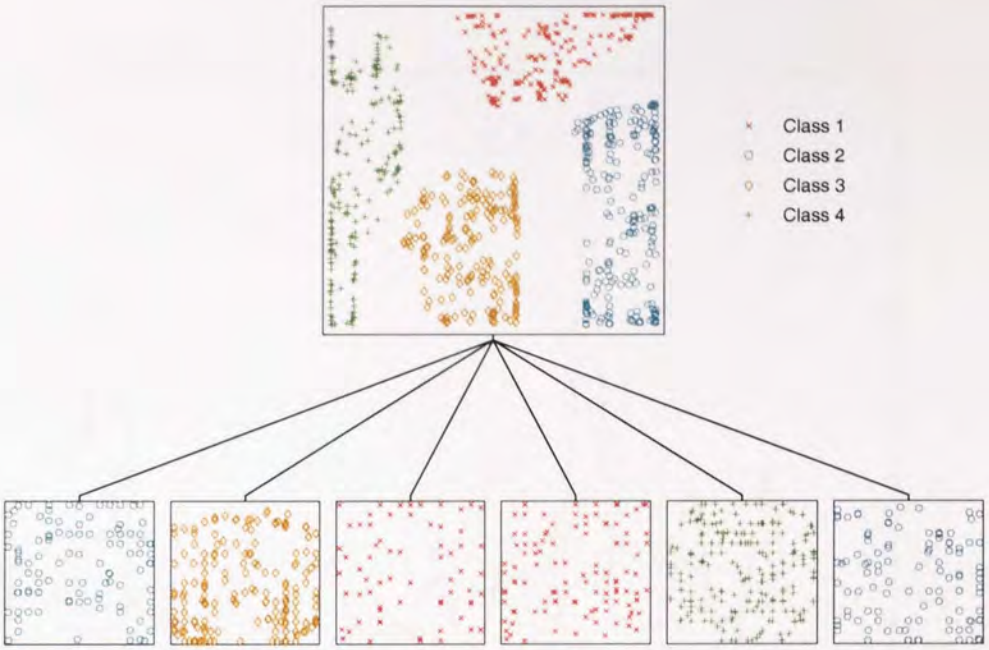


Figure 5.3: Visualisation of the toy data constructed in an unsupervised MML way.

error value at each iteration. When $A_+ = 6$, it has the lowest value of $\jmath(\hat{\boldsymbol{\theta}}(j-1), \zeta)$, consequently it is survived. We stress that there is no contradiction between the number of components (6) in the final mixture of LTMs and the dataset composed of four "humps". There is no driving force in the MML formalism to achieve this and this is not the point of our study. The important thing is that the MML methhod finds a good number of subplots so that the overall probability of the dataset is high (good projections) and the mixture model is not too complex (unnecessarily high number of subplots). In additon, it *automatically* finds appropriate *positions* of the projection manifolds in the data space.

Moreover, we did the experiment with the same conditions, but without the "fresh start". The projection manifold and visualisation plots are presented in Figures 5.5 and 5.6. This time a 7-component mixture was constituted. It suggested that the mixture model can be improved (to a degree) with "fresh start".

For comparison, we carried out an experiment on the same dataset using the BIC criterion. To avoid sensitivity to the initialisation of EM algorithm, we use $K-$means to set the initial conditions. The number of components of candidate mixtures ranges from 2 to 10. Figures 5.7 and 5.8 show the projection manifold and visualisation plots, respectively. In comparison with the MML, the result still looks reasonable, though more components were retained with the BIC. However, we must note that here the BIC has the aid of $K-$means, which gave a good initialisation for this well-separated dataset. When meeting a "messy" dataset, we believe that it will not work so well as this result (An example will be given in section 5.5.1). In addition, as is well known, $K-$means is not a suitable solution for discrete datasets. Thus, we shall not recommend to apply the BIC as an initialisation approach.

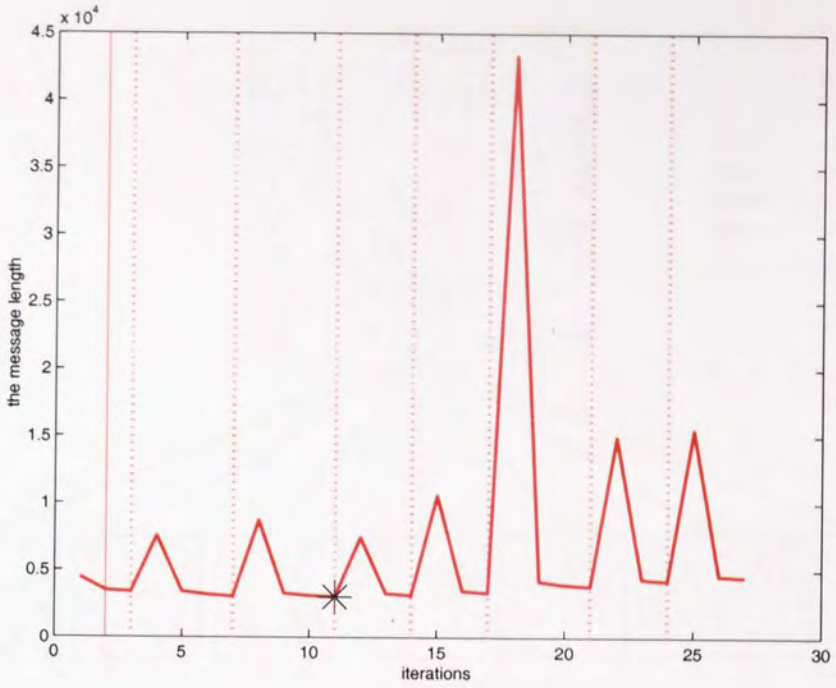


Figure 5.4: Evolution of the cost function. The vertical solid lines signal the annihilation of one component inside the CEM algorithm; the vertical dotted lines indicate the least probable component being forced to zero after the convergence of CEM. The minimum error is shown by a black star.

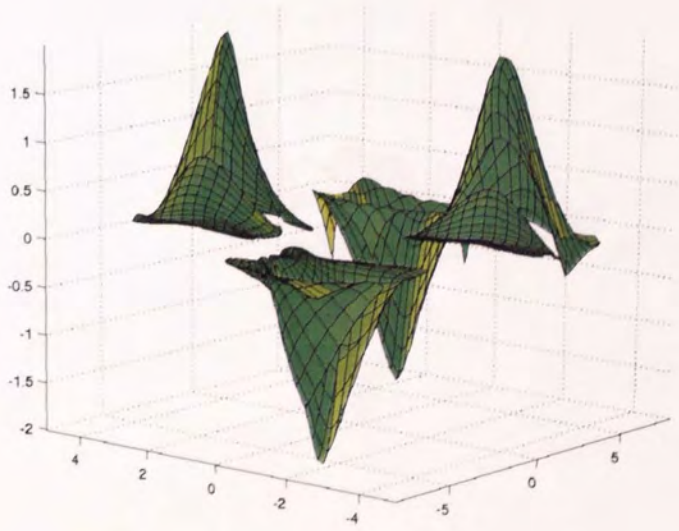


Figure 5.5: Projection manifolds in data space of the second-level LTMs trained on the toy data without a fresh start.

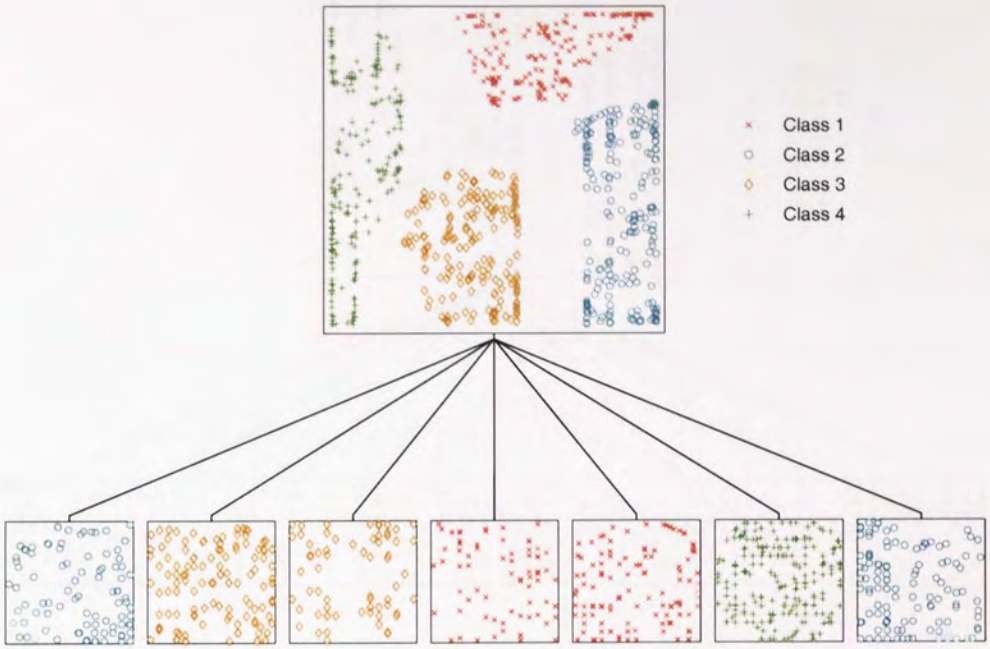


Figure 5.6: Visualisation of the toy data constructed in an unsupervised MML way without a fresh start.

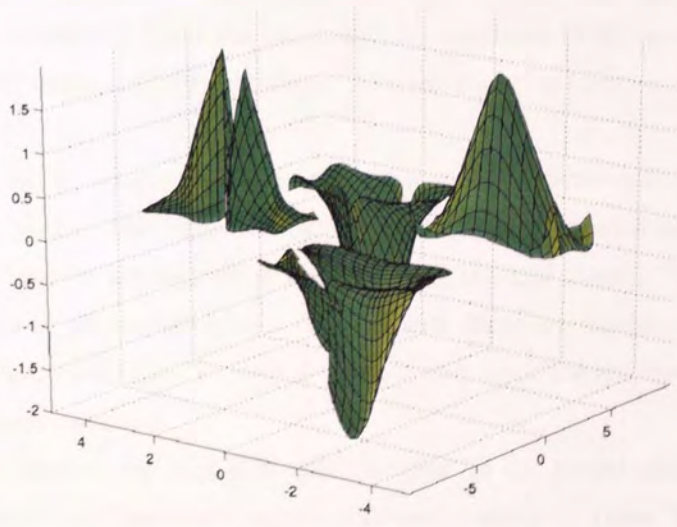


Figure 5.7: Projection manifolds in data space of the second-level LTMs trained on the toy data with BIC initialisation.

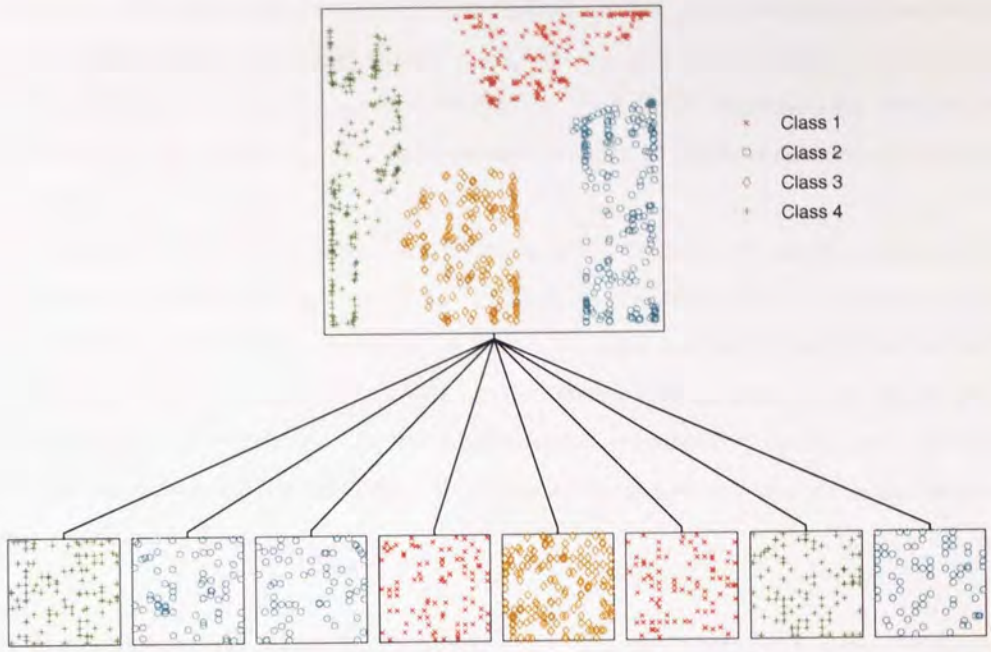


Figure 5.8: Visualisation of the toy data constructed in an unsupervised way with the BIC initialisation.

5.4 Semi-supervised learning of visualisation hierarchies

5.4.1 Semi-supervised visualisation system

The procedure discussed in this section is more complex for nodes on level > 2. In this case, we should consider model responsibilities of the parent nodes for the data points and these are recursively propagated as we incrementally build the hierarchy. So equations (4.25) and (4.24) are used in a hierarchy instead of equation (5.33) for mixtures. Also the equation (4.11) is applied in place of the equation (5.34).

The proposed system for constructing hierarchies of non-linear visualisation plots is similar to the one described in section 4.4. The important difference is that now, given a parent plot, its children are not always constructed in the interactive way by letting the user identify "regions of interest" for the sub-plots. In densely populated higher-level plots with many overlapping projections, this may not be possible. Instead, we let the user decide whether they want the children to be constructed in an interactive or unsupervised way.

In the interactive method, the user clicks the "centres" in the parent plot depending on which regions he is interested in and how many regions of interest there are. These "centres" will be initial mixture models in the deeper level (see section 4.3.5). In the unsupervised method, we use the MML technique to decide the "appropriate" number and approximate position of children LTMs for current (parent) LTM. We collect data points from ζ for which the parent LTM has responsibility higher than

the threshold Δ (in our experiments we set $\Delta = 0.9$). This is a similar case mentioned in section 5.3.2, where all child models arises from the same parent model 'Root', whose responsibility is 1. We then run the MML-based learning of mixtures of LTMs (section 5.3.2) on this reduced dataset. The resulting local mixture is viewed as an *initialisation* for the full EM algorithm for training hierarchies of LTMs (section 4.3.3). This way, the "appropriate" number of LTMs is determined along with their initial locations.

When we consider the log-likelihood as a criterion in the training process of a hierarchical system, what we obtain to update weight matrix is equation (4.27), where $R_{\mathcal{M}}$ is to simply rescale $R_{kn}^{\mathcal{M}}$ (equation (2.72)) with $P(\mathcal{M}|\mathbf{t}_n)$ (equation (4.24)). We cannot directly apply this for mixtures at deeper levels of the hierarchy, since the MML would mean it was no longer a quadratic problem. To keep the principled framework for the whole hierarchical visualisation system, we apply the MML technique just for sub-model initialisation. As for the training process, the procedure keeps the same as the one described in 4.3.3.

5.5 Summary of the hierarchical visualisation procedure

We have developed software, based on NETLAB, to implement the hierarchical visualisation. The basic procedure can be summarised as follows:

Step 1. Input normalisation.

A simple linear rescaling is useful when observed variables have significantly different values. For example, when we dealt with the screen data, for each variable t_d, we computed its mean $\bar{t}_d$ and variance σ_d^2 with respect to the whole dataset. Then a set of re-scaled variables was given by

$$\tilde{t}_d = \frac{t_d - \bar{t}_d}{\sigma_d}. \tag{5.36}$$

A more complex rescaling can be found in (Bishop, 1995) (Chapter 8).

Step 2. Initialise structural parameters.

A set of structural parameters have to be initialised before the training process. Consider a two-dimensional latent space. Key parameters are the shape of latent space, RBF centres' layout, width of RBF kernels. For example, for the experiments in this thesis, we chose the regular grid for both *Shape of latent space* and *RBF centres' layout*, and selected the mean of the minimum distance from each centre to its neighbour as *Width of RBF kernels*. Several different choices can be found in NETLAB. More details on the selection of these parameters have been discussed in section 2.5.2.

Step 3. Train the whole dataset using one single LTM.

Practically, we considered criteria for stopping the training process as follows:

- Firstly, to observe whether or not the error function is convergent;

  Usually for datasets used in this thesis, it converged in 60 to 120 iterations.

- Then to observe whether or not projections of data points change obviously in the plot.

For some discrete datasets, even when the error function is approaching convergence, the visualisation plot can still change a lot compared with one obtained after a few more training iterations.

Step 4. Select a mode for determining regions of interest.

One can analyse visualisation plots with auxiliary variables, e.g. both magnification factors and curvatures. If clusters are separated clearly, then the user can select the centres using the *interactive* mode. Otherwise, by choosing the *automatic* mode, the minimum-message-length-based method will be used. In this case, the user will suggest a maximum number of components A_{max} which they would like to consider. For datasets used in this thesis, we chose $A_{max} = 10$. One can also set a minimum number of components A_{min}. This means that the user just wants to consider a reasonable number of components between A_{max} and A_{min}.

Step 5. Train sub-models together and setting up plots at the next level.

Step 6. Repeat step 4 and 5 for each sub-model.

Some practical considerations can be found in section 4.3.5.

Step 7. Show final results for analysis.

One will obtain a hierarchical projections plot and a magnification factors plot. For continuous data, we also have a curvature plot to support data analysis.

5.5.1 Experimental results

In this section we illustrate the semi-supervised hierarchical LTM visualisation algorithm on four "real-world" data collections.

Structural parameter selection was the same as in section 4.5. Note that, as mentioned in section 4.4 in the interactive mode, the "centres" of the regions of interest are shown as circles labeled by numbers. These numbers determine the order of the corresponding child LTM subplots from left to right.

Image segmentation data

As the first example we visualise the image segmentation dataset. The final hierarchical visualisation plot of GTMs can be seen in Figure 5.9. The *Root* plot contains clusters of overlapping projections. Six plots at the second level were constructed using the unsupervised MML technique ($A_{max} = 10$). Note that the second-level LTMs already separate the four classes fairly well and are readable enough to be analysed further in the interactive mode. For example, we selected three, two and two "centres" respectively for regions of interest (shown as circles) in the first, third and fourth level-two plots.

Furthermore, we repeated the experiment with the BIC initialisation. Again, we applied $K-$means to initialise sub-models. The range of the number of components was from 2 to 10. Finally, a 10-component mixture was constructed. By comparison with Figure 5.9, segments at the second level are dispersed. Because the dataset was not well-separated in the *Root* plot, the BIC did not obtain a

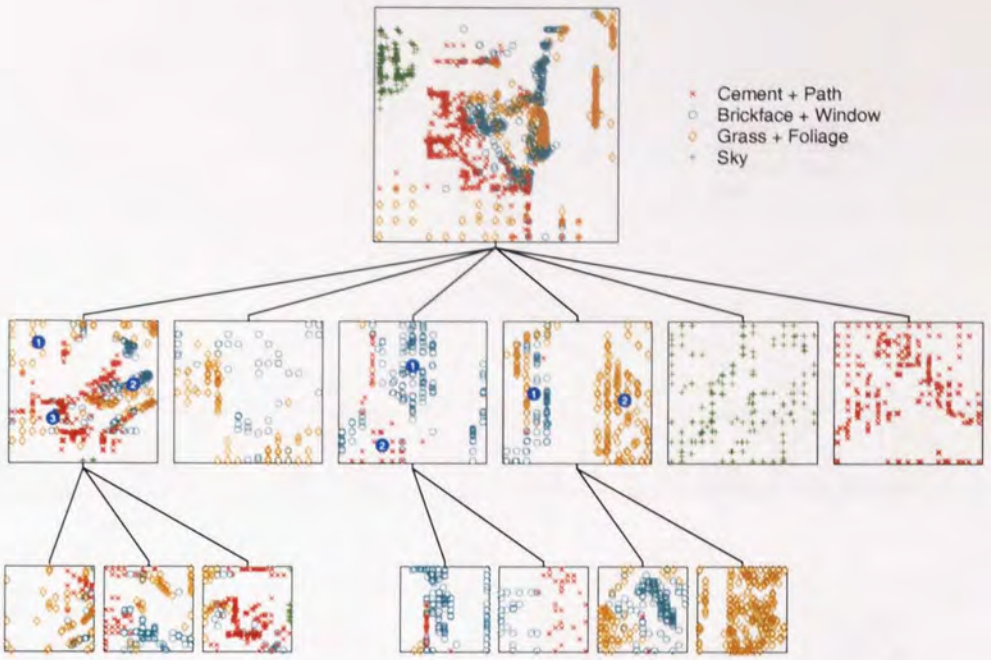


Figure 5.9: Hierarchical visualisation of the image segmentation data constructed in a semi-interactive way with the MML initialisation.

better aid from $K-$means. It indicates that initialisation with the MML is more effective.

HTS dataset

In this experiment we visualised the HTS dataset using a hierarchy of GTMs. A semi-supervised hierarchy of GTMs down to level 3 was trained on this set and the final visualisation plot is shown in Figure 5.11. 3 plots at the second level were built by using the unsupervised MML technique ($A_{max} = 10$). They contain readable groups. We chose the regions of interest to further refine the data in the third level.

Document dataset

Since our system is based on the LTM, it can deal with discrete dataset. To account for the binary encoding, a Bernoulli noise model was employed.

The visualisation plot generated in a semi-interactive way is shown in Figure 5.12. The 'Root' is extremely densely populated with highly overlapping data projections. After using the unsupervised MML technique ($A_{max} = 10$), a 4-component mixture of LTMs was obtained on the second level. Sub-clusters in these four level-two plots are decipherable. The user can now choose more detailed regions of interest by using the interactive mode.

As in chapter 4, this system also includes the child-modulated ancestor plot technique, which can visualise the regions captured by a particular child LTM $\mathcal{M}$. This improves the user's understanding

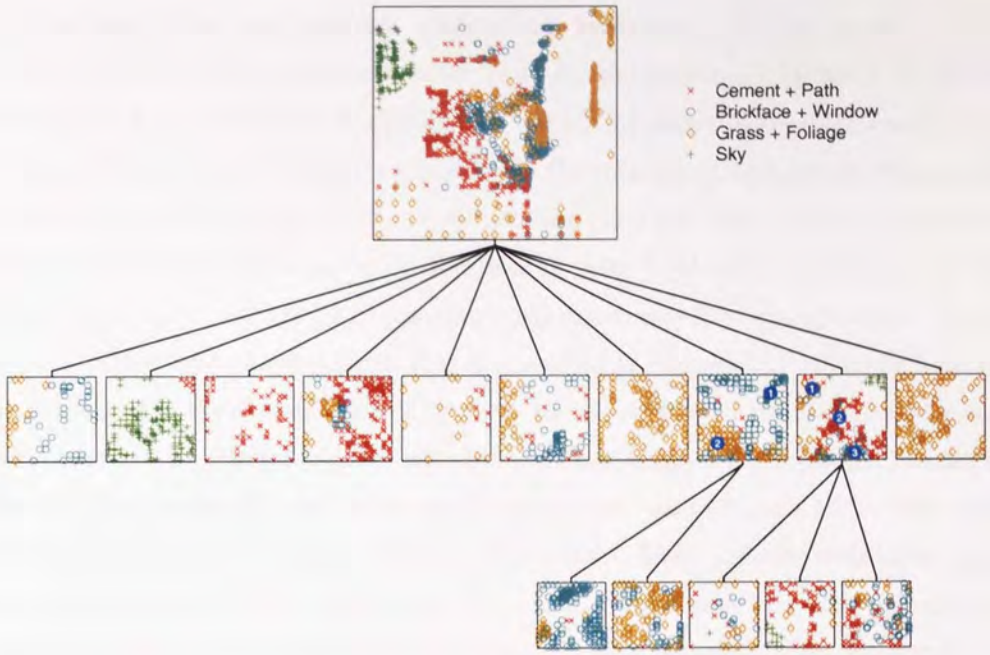


Figure 5.10: Hierarchical visualisation of the image segmentation data constructed in a semi-interactive way with a BIC initialisation.

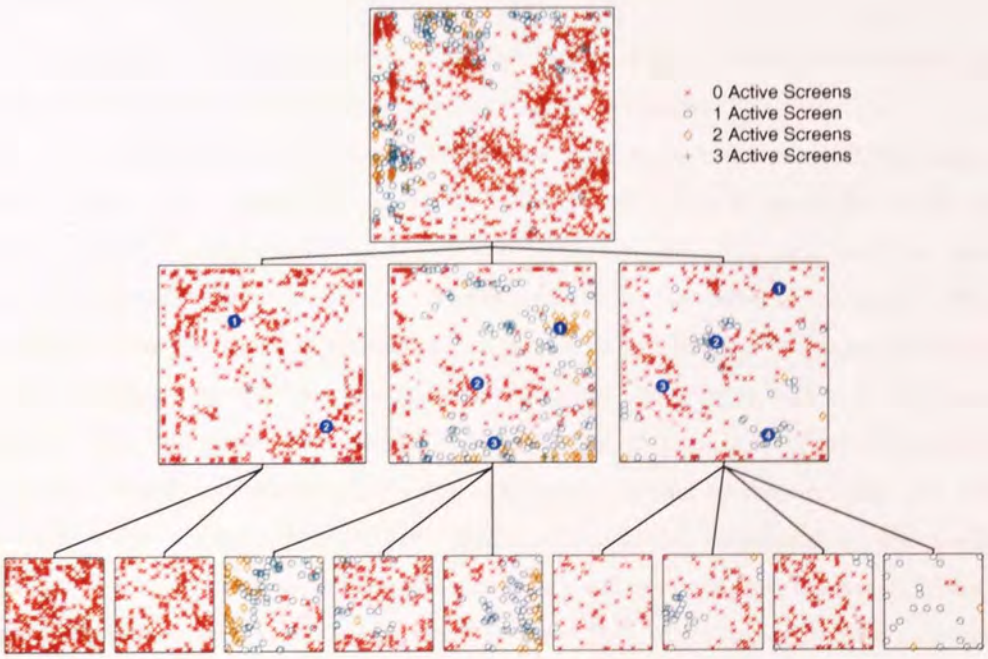


Figure 5.11: Hierarchical visualisation of the HTS data constructed in a semi-interactive way.

of the relationships among sub-plots in the visualisation hierarchy. In Figure 5.13, we highlight the visualisation plots which include the data points captured by the first model at the fourth level. It indicates that most points from the topic 'sci.space' are explained in this sub-model.

Figure 5.14 shows the magnification factor plots for the projection hierarchy of the document dataset in Figure 5.12. There is a dark band slightly left of centre in the 11th level-3 model. The band divides different topics in the data space. From the corresponding model in Figure 5.12, we see that the left part mostly involves topic 'talk.politics.misc', and the right contains a mixture of topics.

For a detailed analysis, we focus on the fourth level-three LTM model in Figure 5.14. The corresponding projection plot in Figure 5.12 contained only documents from a single topic, 'sci.space'. An enlarged view of the magnification factor plot is presented in Figure 5.15. It can be seen that there is a dark band around the diagonal line of the plot. Hence, we infer that documents in either side of the band correspond to different clusters and that a change of sub-topic happens. The list of 5 most probable dictionary words for each latent space centre is shown in Figure 5.16. With reference to Figure 5.15, two clusters are found in corresponding regions. Key words for each latent space centre inside the region bounded by the solid border are completely the same and have the same orderings. They appear to refer to documents relating to space shuttle launches, while key words inside the region with the dashed border seem to be associated with articles concerning space orbits.

Although the magnification factor plots may help us, we must indicate that it may not be useful for comparing the absolute clustering compactness across the plots or levels, since the magnification factor plots is scaled (i.e. normalised).

Yeast dataset

In the last experiment we visualise the yeast dataset in Figure 5.17, and we demonstrate application of the unsupervised MML technique at a lower level in the hierarchy.

We trained a four-level hierarchy of LTMs on the yeast data and the resulting projections are displayed in Figure 5.17. Again, the *Root* plot looks 'messy'. Two plots at the second level were constructed using the unsupervised MML technique ($A_{max} = 10$). The first level-two plot is clear enough for the user to select the centres in the interactive mode (as shown in the figure). We used the MML algorithm as an initialisation technique for constructing child plots of the second level-two plot ($A_{max} = 5$). Two resulting child plots included readable clusters. Figure 5.18 is the child-modulated ancestor plot. The data points captured by the first model at the 4th-level are highlighted. Again this figure tell us which data points in the parent plot are explained by the child plot. As viewed, the points grouped are mostly from class *ME3* and their positions in the corresponding plots are displayed.

Figure 5.19 presents a result with the same conditions, but without the "fresh start". We see *ME3* class were finally divided into two sub-clusters at the level four, while this class were more reasonably grouped into one cluster at the fourth level in Figure 5.17. It suggests that it can give a better model with a "fresh start".

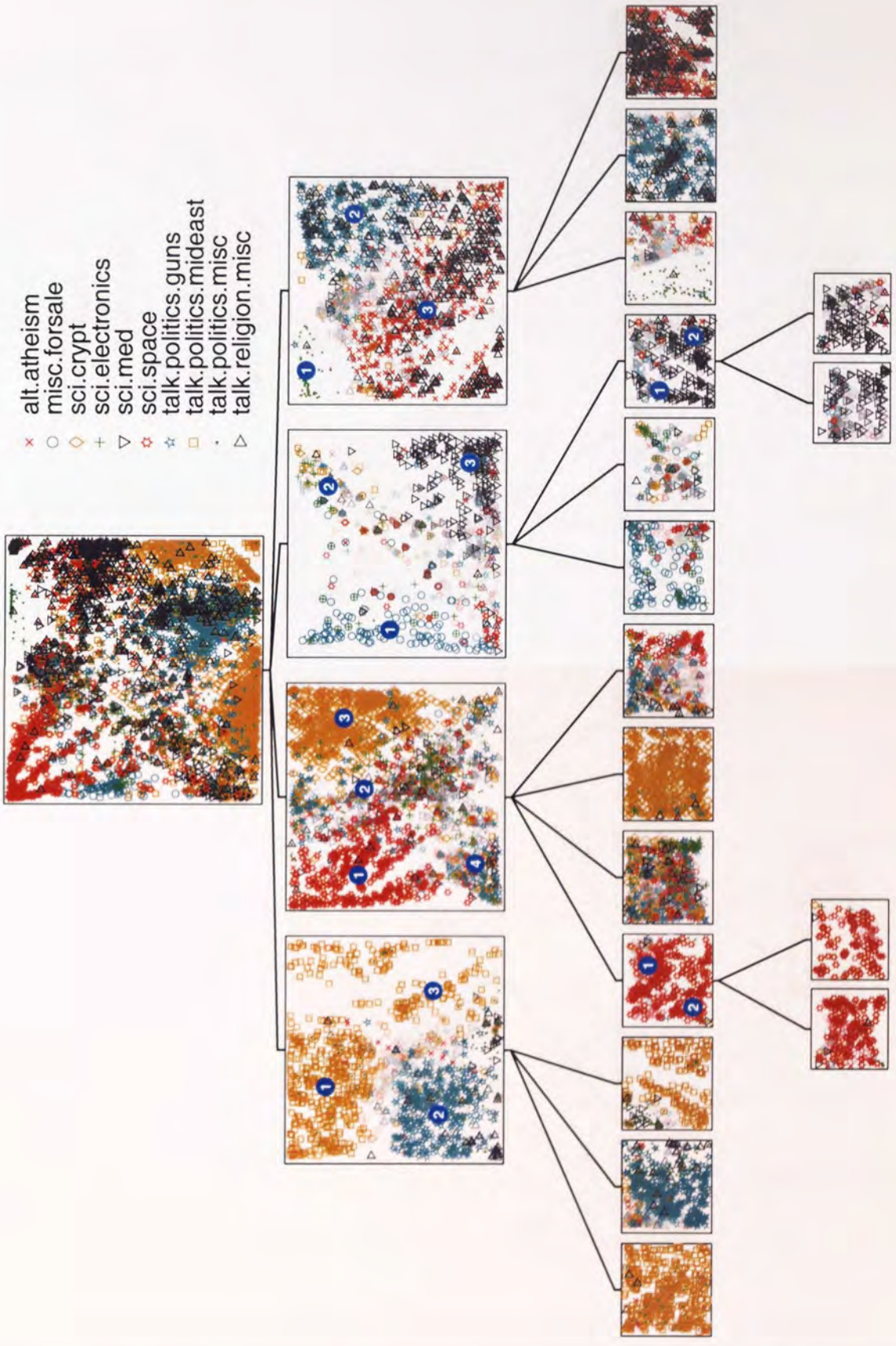


Figure 5.12: Hierarchical visualisation of the document data constructed in a semi-interactive way.

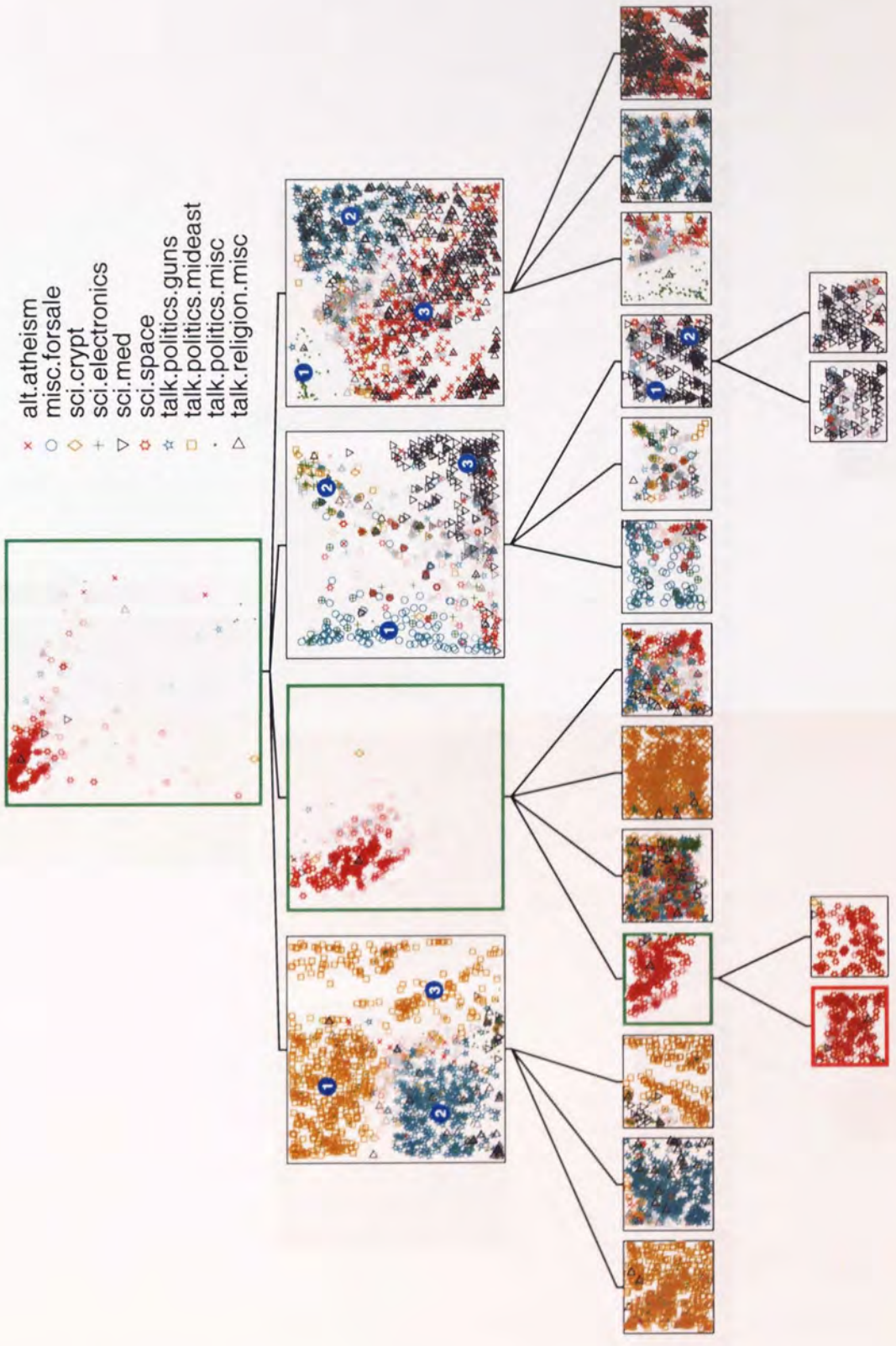


Figure 5.13: Hierarchical visualisation of the document data constructed in a semi-interactive way. The set of points captured by the first LTM at level 4 of the hierarchy is highlighted in the visualisation plots of all its ancestors.

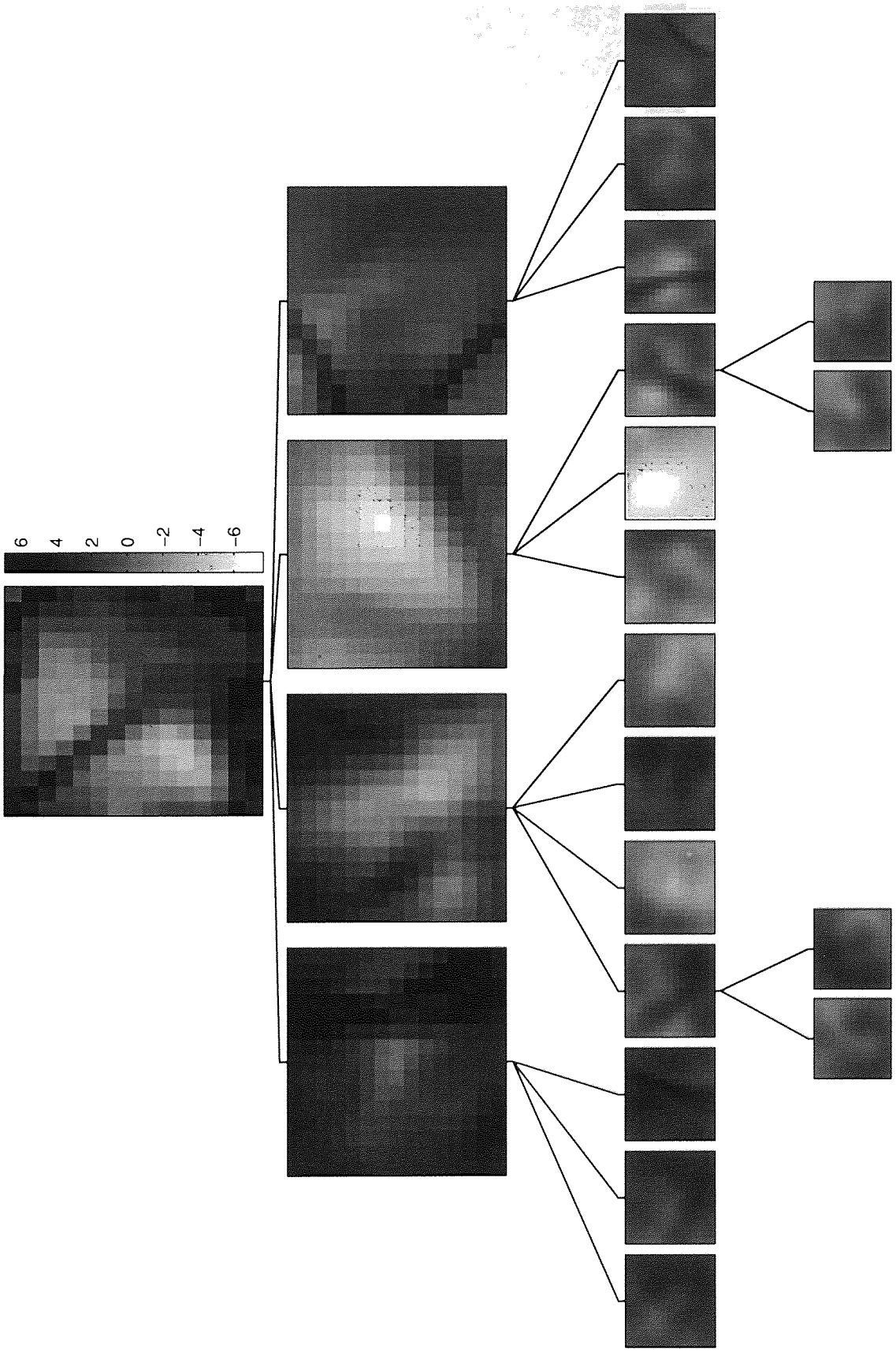


Figure 5.14: Plots of magnification factors in the hierarchy of LTMs fitted on the document data.

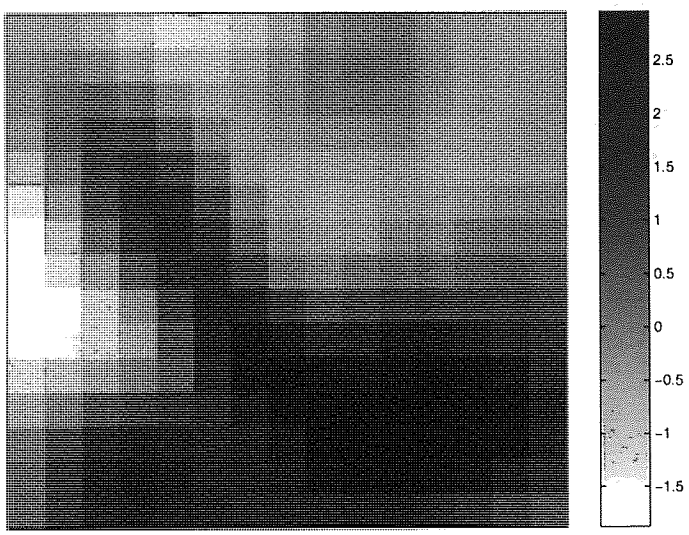


Figure 5.15: A visualisation plot of magnification factors for a LTM.

5.6 Discussion

In this chapter we have presented a semi-supervised hierarchical visualisation system. The proposed system gives the user a choice of initialising the child plots of the current plot in either *interactive*, or *automatic* mode. In the automatic manner, an approach based on the MML is employed and the initialisation has improved with a "fresh start". We have further showed how to use magnification factor plots to analyse the dataset and find out sub-clusters.

Compared with other techniques for the model order selection, this approach seamlessly combines model selection criterion into an EM-based training algorithm. It is particularly useful when user has no idea how to choose the areas of interest due to highly overlapping dense data projections.

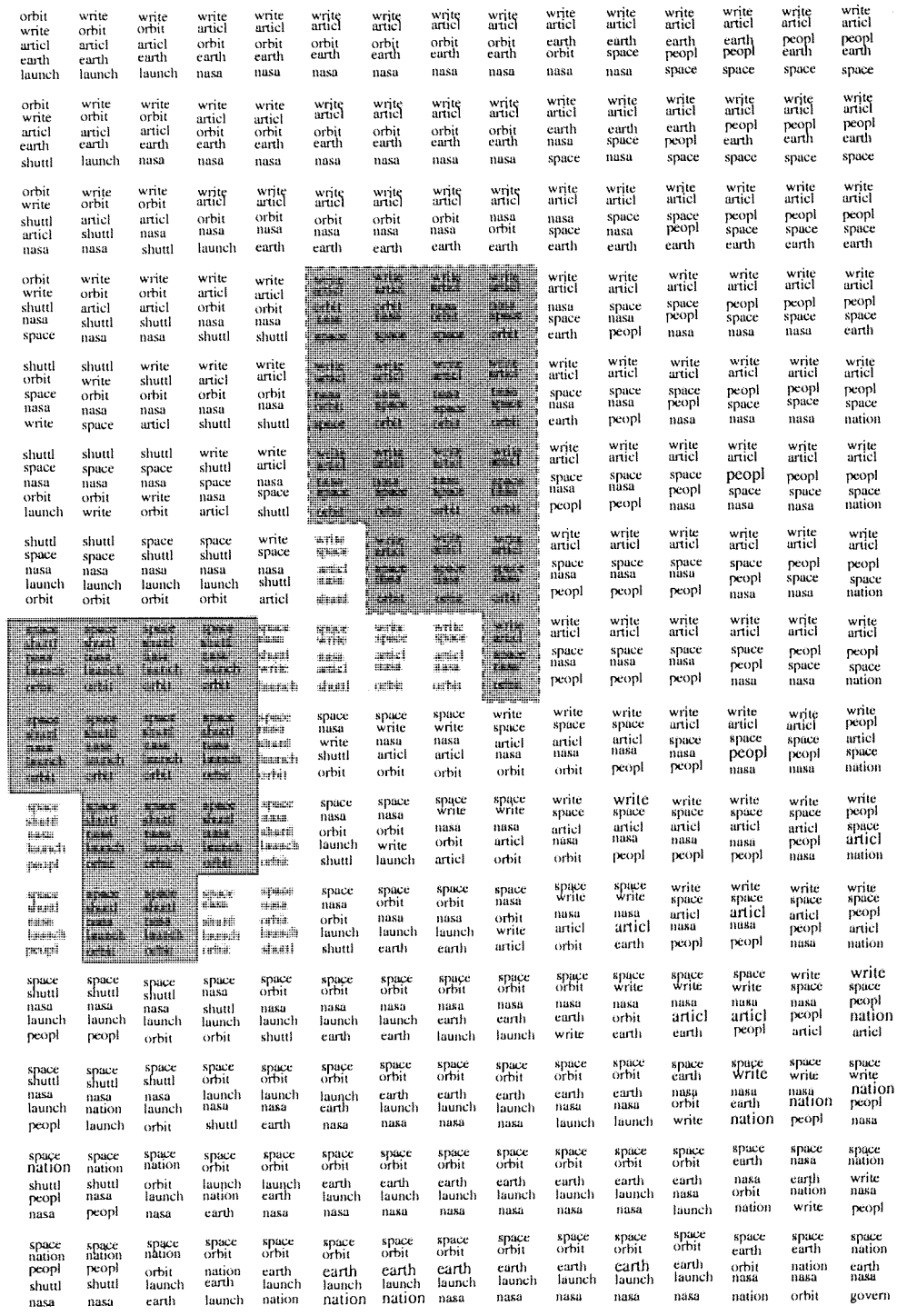


Figure 5.16: The most probable words formed in each of the 15 by 15 latent grid points by the binomial latent trait model.

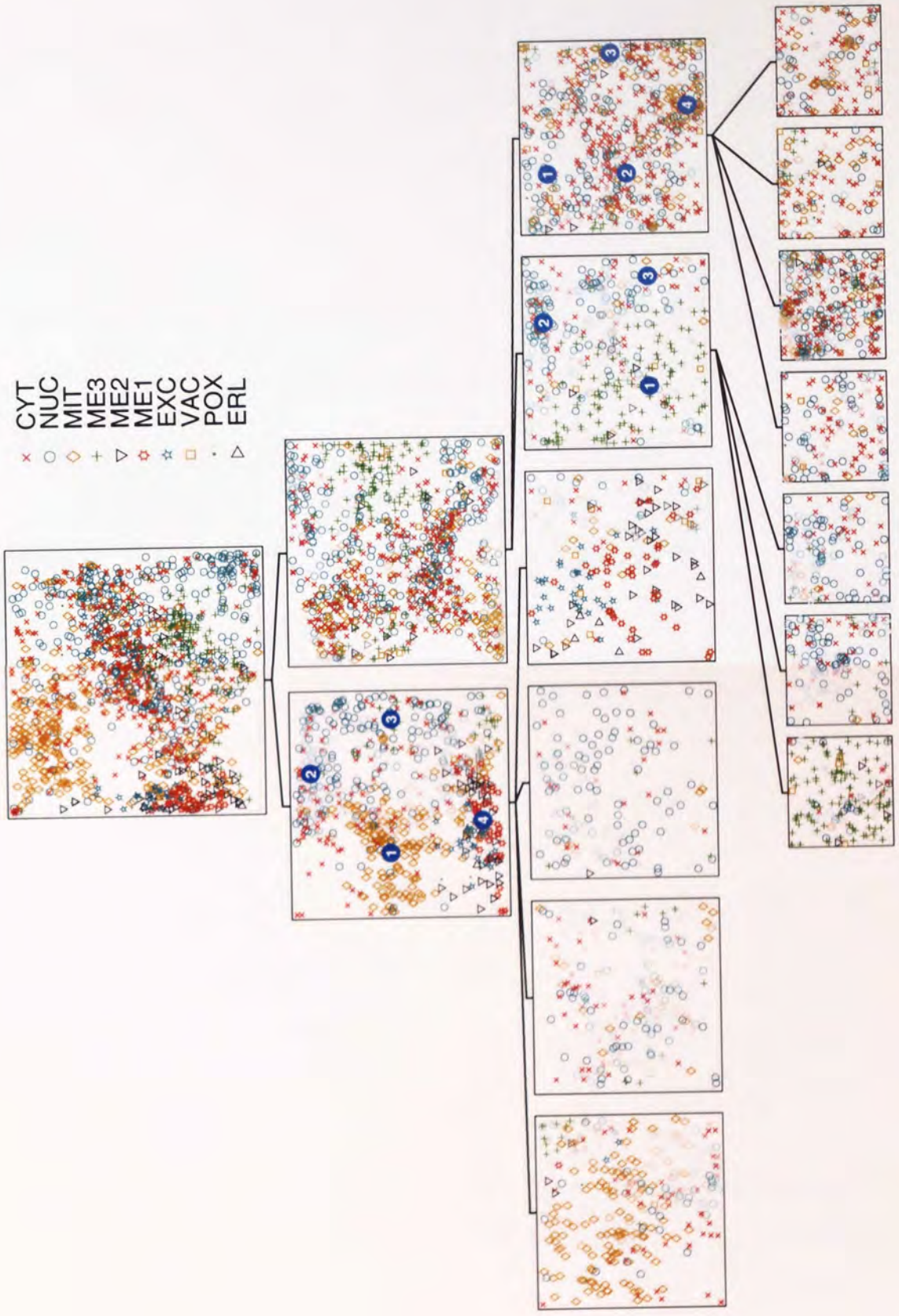


Figure 5.17: Hierarchical visualisation of the yeast data constructed in a semi-interactive way.

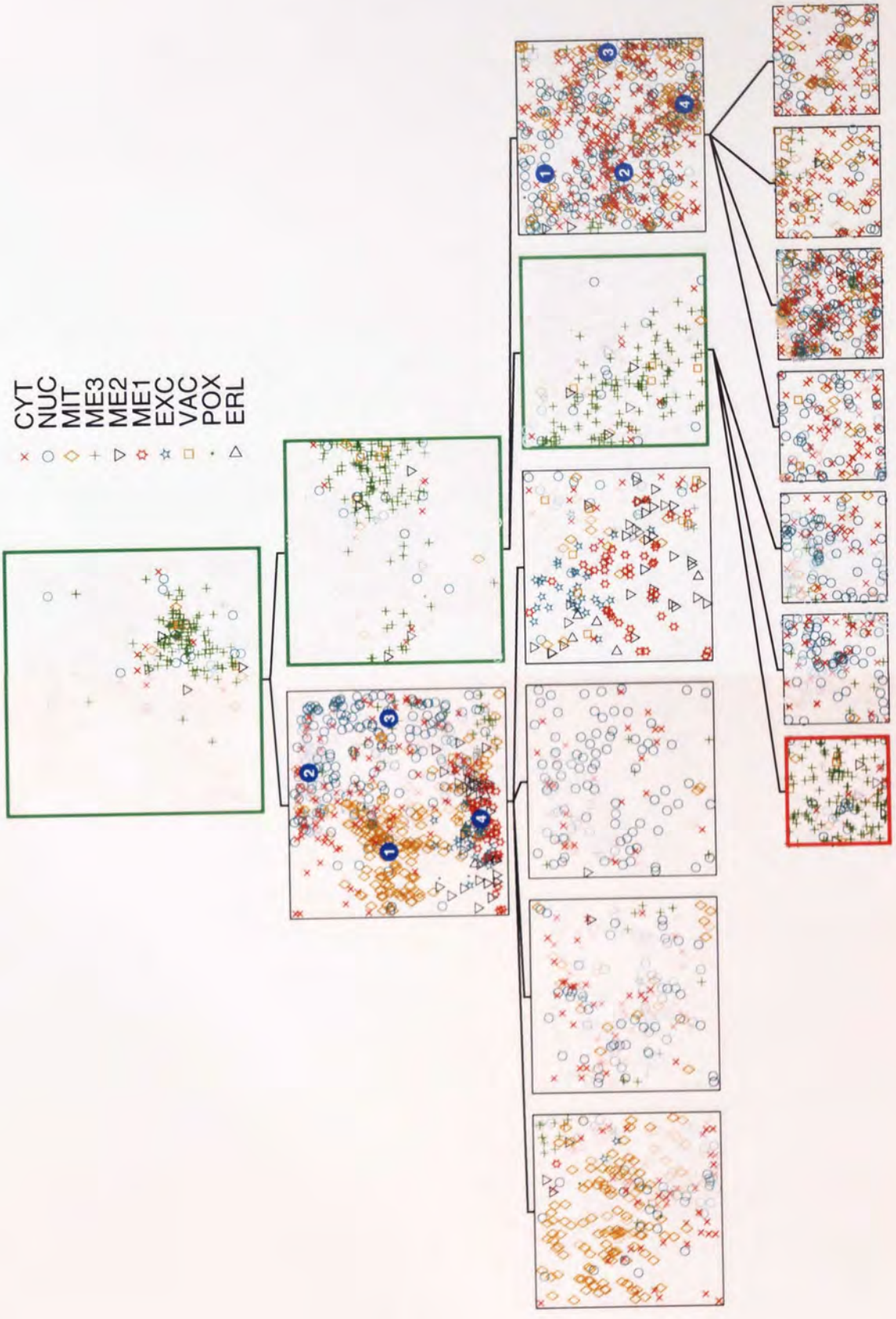


Figure 5.18: Hierarchical visualisation of the yeast data constructed in a semi-interactive way. The set of points captured by the fourth LTM at level 4 of the hierarchy is highlighted in the visualisation plots of all its ancestors.

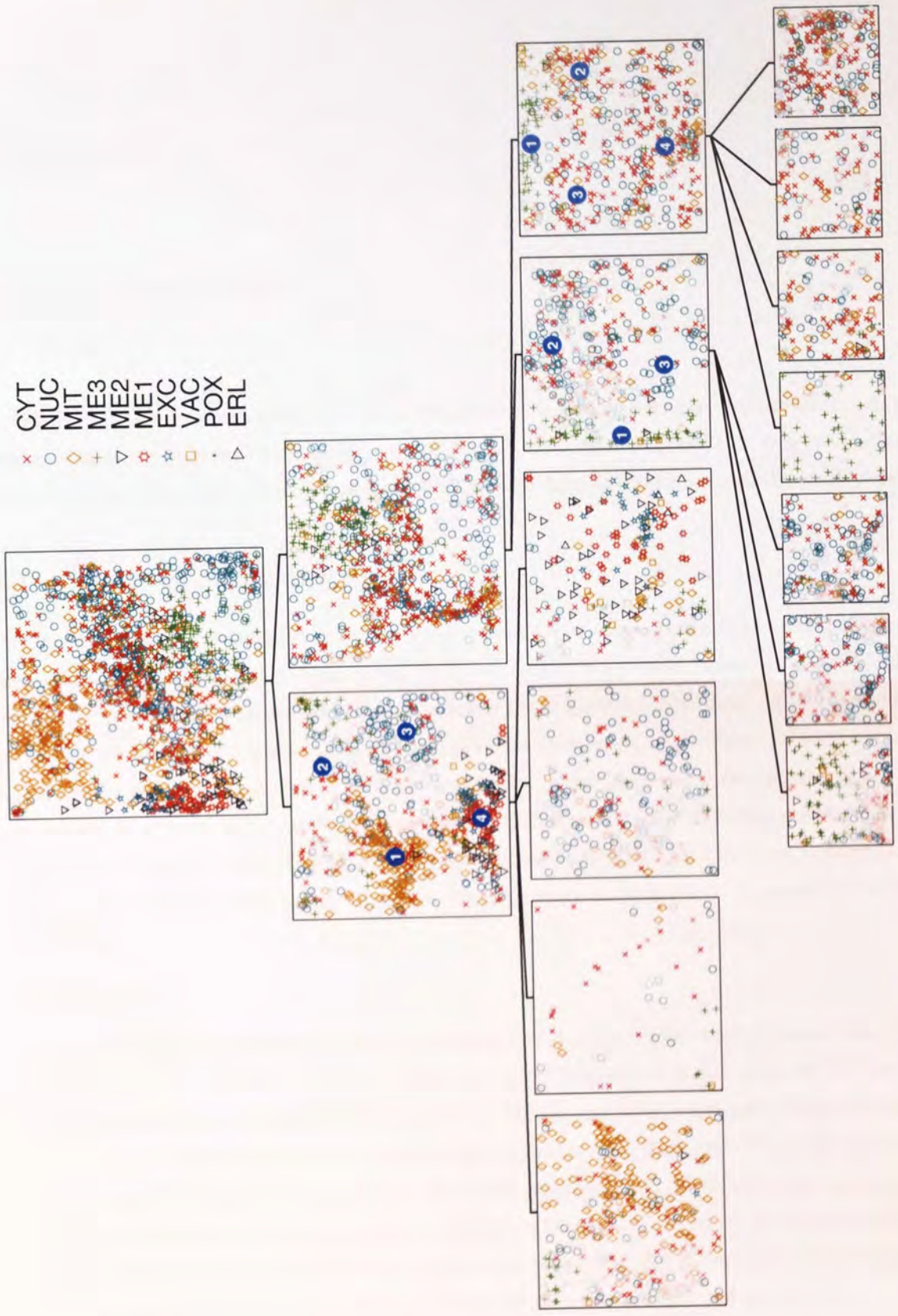


Figure 5.19: Hierarchical visualisation of the yeast data constructed in a semi-interactive way without a fresh start.

Chapter 6

Conclusions

In this final chapter we summarise the work described in previous chapters. Our method can be employed in many domains. As examples, we sketch out what can be obtained in Web mining and microarray data expression. Finally, we discuss some development directions and end the whole thesis.

6.1 Chapter summary

The research described in this thesis focuses on searching for a viable approach to visualise a large quantity of data. The method we applied and further developed is a general framework of a hierarchy constructed from latent trait models, which transforms the original observed data, either continuous or discrete, to a lower dimensional space by revealing the structure in the dataset as accurately as possible. The clusters represented in each projection plot in the hierarchical tree provide a key to understanding the relationship between data points in the data space. The application of this type of visualisation techniques to the large HTS dataset has revealed useful information to our collaborators at Pfizer. Moreover, this approach can be used in other domains, e.g. as shown in this thesis, document data mining.

- **Chapter 2**

  In this chapter, we reviewed the general framework of mixture models and EM algorithms, and investigated several latent variable models, such as PPCA (which recasts classical PCA into a framework of density models), GTM and LTM. We also discussed Neuroscale, which is a non-linear topographical projection. From experimental results using these approaches, we concluded that both GTM and LTM, involving a non-linear mapping from the latent space to the data space, are effective tools for visualising the large HTS dataset. However, it is known that a single plot is not enough to capture all important information from a more complex dataset. This motivated us to further develop a hierarchical system based on LTM and GTM.

- **Chapter 3**

In this chapter, we described extensions to the GTM in two directions.

1. Since the GTM forms a smooth manifold, it allows us to use differential geometry as an analysis tool. We studied curvatures of the manifold based on second order derivatives. We visualised the curvatures' direction and magnitude, which may indicate how curved the manifold is when embedded in the data space.
2. One practical difficulty in the use of the GTM arises from incomplete data points in a dataset. For visualisation purposes, we integrated the estimation of missing values into the model training process using an EM algorithm. Better estimates of missing values can be achieved by using additional class information as a constraint.

- **Chapter 4**

  In this chapter, the LTM is extended to a hierarchical structure with nodes corresponding to local models. We integrated projection visualisation, magnification factors, curvatures and the point-list into our hierarchical visualisation system. In *interactive* mode the user is responsible for choosing regions of interest, which are used for parameter initialisation of the mixture of LTMs in the next level. We improved the standard initialisation so as to deal with discrete data. We illustrated this hierarchical system on several data sets with different noise models from exponential families. Experimental results suggest the system is helps to capture more interesting information in the deeper levels. Especially, the results we obtained to the HTS data is useful to Pfizer's Medicinal Chemists.

- **Chapter 5**

  In this chapter, we further improved our hierarchical visualisation system by providing an *automatic* initialisation mode based on the MML principle. Experimental results suggested that this mode is useful when facing a heavily overlapping and messy plot, for which a reasonable number of components and their positions can be automatically determined by the algorithm. By comparison with the BIC, an alternative method of model selection, our approach can provide a better separation.

6.2 Applications

Our approach can be used for analysing large datasets and searching for unexpected relationships in the data. The resulting techniques typically construct several local models that represent the structure of the data in an easily understandable way. Usually, the data represented in one plot in the deeper levels of the hierarchy have more similarities than at shallower levels, since the members of one cluster different from one another as little as possible (Spath, 1980). This can help us to find similar records from databases and search for documents which describe a similar topic.

The method can be applied in many industrial applications, such as medical genetics, text databases, and so on. In the area of language technology people are interested in finding interrelated stories from

different news feeds. Using our approach, news text can be classified into topic classes. Then the interconnected chains of news corresponding to event reports can be recognised. In this section, we discuss two more possible applications.

- Web mining.

  As the Web becomes more and more popular in the whole world today, the amount of information on it is growing extremely quickly. A survey on web mining research can be seen in (Kosala and Blockeel, 2000).

  Usually a user inputs a set of keywords using a searching engine, which will respond with a list of pages according to the similarity to the keywords. One of the key tasks of Web mining is Web document classification or categorization, which can be used for indexing (Kosala and Blockeel, 2000).

  Here we consider a case ignoring the sequence in which the words appear and the document has no structure, e.g. HTML tag, in it.

  By using a hierarchical latent trait model, the original document vector representation is transformed into two-dimensional space. What we can find are

  - Similar documents in the same topic but with different sub-topics;
  - Similar documents with lots of similar terms in the same sub-topic.

- Microarray data.

  With current revolutionary developments in the life sciences, scientists are studying genes with immense zeal and interest. DNA microarrays, also known as DNA or gene chips, allow scientists to measure the expression level of thousands of genes from a single biological sample on one microchip (Jagota, 2001).

  A microarray or chip contains thousands of spots. Each spot includes thousands to millions of copies of a single DNA strand representing a gene. These chips work by following a property of DNA and RNA: *complementary base pairing*.

  If a DNA molecule from a tissue sample binds to a gene on a spot of the chip, the researcher can infer that the molecule from the sample has the complementary sequence of that gene's. The DNA or RNA molecules are added fluorescent tag or other label, which can be detected by a scanner. Once a chip has been scanned, a computer converts the raw data into a color-codded readout (Friend and Stoughton, 2002).

  Microarray data is often expressed as an $N \times D$ matrix $\mathbf{T}$ where N are number of genes, D biological samples and t_{nd} denotes the expression level of gene n in sample d.

  Note that usually before microarray data is stored in a database, it may have undergone a number of transformations. So it is important for the user to know what transformations have been done to the data. What we can discover from microarray data using hierarchical visualisation are

- Genes with similar expression patterns over all samples.

  It is useful to find outliers and any genes whose function is similar to a known gene.

  Since there are thousands of genes, it may generate a highly messy plot. If so, the *automatic* mode based on MML will be of help to further cluster the similar points.

- Samples with similar expression patterns.

  Scientists may also want to know whether or not a sample is a cancerous, or which subtype of this kind of cancer it is, and so on. At this time, we consider D samples as inputs, while N genes as variables. There is a challenge here since variables involve with thousands of genes. However, this difficulty can be overcome by using PCA to reduce dimensionality first and transform data points onto a smaller number of principal components as new dataset to further visualise.

6.3 Discussion and open questions

The hierarchical visualisation algorithm we have proposed can be further developed in several directions.

1. Extensions to the LTM.

   - Curvature for latent trait model.

     In this thesis, we just consider local directional curvatures of the GTM manifolds. As a natural development, one can compute curvatures on the manifold formed by the LTM. In that case, the image of the line $\mathbf{x}(b)$ generates a curve in the projection manifold governed by the link function (see section 2.6), so the equation (3.4) can be re-written as

     $$\mu(b) = g(\boldsymbol{\theta}_{\mathcal{M}} \boldsymbol{\Phi}(\mathbf{x}_0 + b\mathbf{h})). \tag{6.1}$$

     Then the *embedding curvature* $\ddot{\mu}^{\perp}(0)$ can be calculated using the same way as for the GTM.

   - Incorporating missing values into the EM algorithm for the LTM model

     Here we discuss the case of the Bernoulli distribution. As we did for the GTM, to incorporate missing data we must calculate the appropriate expectations of the sufficient statistics in the $E-$step. For the Bernoulli mixture, they are $\langle z_{kn} | \mathbf{t}_n^o, \boldsymbol{\theta}_k \rangle$ given by R_{kn} (equation (2.72)) computed over the observed $\mathbf{t}_n$; and $\langle z_{kn} \mathbf{t}_n^m | \mathbf{t}_n^o, \boldsymbol{\theta}_k \rangle$ given by $R_{kn} \boldsymbol{\mu}_k^m$, where $\boldsymbol{\mu}_k^m$ is the probability of the kth variable being 1 in the mth dimension. The $M-$step uses these expectations to update equation (2.75). The extension to the multinomial distribution is straightforward.

2. Temporal data.

   The standard GTM described in this thesis assumes that the data is generated as an independent, identically distributed sample. However, in some applications this assumption is not valid. For

temporal data, an extension has been proposed by incorporating a hidden Markov model for transitions in the latent space (Bishop *et al.*, 1997). Since the model is trained using an EM algorithm, it is straightforward to develop it in a hierarchy system. In particular, the hierarchical visualisation plot can be used to compare different states and analyse where and how much changes that happened between two steps.

3. Speeding up training

   The most computationally consuming part of the training process is in the second level, where a mixture of LTMs are trained with the whole dataset (see section 4.3.5). One way to resolve this problem is to speed up the training procedure of each LTM. As mentioned by Bishop *et al.* (1998), incremental learning (Neal and Hinton, 1999) can be used for training, in which case, instead of doing a full $E-$step, a suitable fraction of the total number of data points is chosen to calculate the corresponding responsibilities. In this way there is a possibility that the algorithm would converge faster because the model will be updated for a small fraction rather than having to wait for a full $E-$step over all data points. In addition, in the $M-$step, we adopted a batch gradient descent technique (see section 2.6). However, a more efficient method may to employ scaled conjugate gradients or other non-linear optimisation techniques (Nabney, 2001).

6.4 Conclusion

The semi-supervised visualisation hierarchy provides a method for modelling either continuous or discrete data with non-linear structures in high-dimensional spaces. As has been exemplified in this thesis, the important application of our work is visualisation of complex high-dimensional data in a hierarchical tree. The possibility of computing magnification factors and directional curvatures and incorporating them in visualisation plots, may make visualisation plots easier to interpret.

Appendix A

Learning From Incomplete Data

For the GTM (Bishop *et al.*, 1998), the negative log likelihood function is

$$E = -\sum_{n=1}^{N} \ln \frac{1}{\mathrm{K}} \sum_{k=1}^{K} p(\mathbf{t}_n|\mathbf{x}_k, \mathbf{W}, \mathbf{\Sigma}),$$

$$\begin{aligned} p(\mathbf{t}_n|\mathbf{x}_k, \mathbf{W}, \Sigma) &= \mathcal{N}(\mathbf{y}(\mathbf{x}_k; \mathbf{W}), \mathbf{\Sigma}) && \text{(A.1)} \\ &= \frac{1}{(2\pi)^{D/2} \mid \Sigma_k \mid^{1/2}} \exp\{-\frac{1}{2}(\mathbf{t}_n - \mathbf{y}_k)^{\mathrm{T}} \Sigma_k^{-1} (\mathbf{t}_n - \mathbf{y}_k)\} && \text{(A.2)} \\ &= p_{kn}. && \text{(A.3)} \end{aligned}$$

Let

$$d_{kn}^2(\mathbf{t}) = (\mathbf{t}_n - \mathbf{y}_k)^{\mathrm{T}} \Sigma_k^{-1} (\mathbf{t}_n - \mathbf{y}_k).$$

We use m and o to denote subvectors and submatrices of the outputs $\mathbf{y}$ matching the missing and observed components of the data.

$$\mathbf{t}_n = \begin{pmatrix} \mathbf{t}_n^o \\ \mathbf{t}_n^m \end{pmatrix}; \quad \mathbf{y}_k = \begin{pmatrix} \mathbf{y}_k^o \\ \mathbf{y}_k^m \end{pmatrix};$$

$$d_{kn}^2(\mathbf{t}) = \begin{pmatrix} \mathbf{t}_n^o - \mathbf{y}_k^o \\ \mathbf{t}_n^m - \mathbf{y}_k^m \end{pmatrix}^{\mathbf{T}} \begin{pmatrix} \Sigma^{-1,oo} & \Sigma^{-1,om} \\ \Sigma^{-1,mo} & \Sigma^{-1,mm} \end{pmatrix} \begin{pmatrix} \mathbf{t}_n^o - \mathbf{y}_k^o \\ \mathbf{t}_n^m - \mathbf{y}_k^m \end{pmatrix}.$$

The form of the covariance matrix is often constrained to be diagonal, and for the GTM, it is further constrained to be spherical with variance is β^{-1}. Thus the covariance matrix has this form:

$$\Sigma_k^{-1,oo} = \beta \mathbf{I}^{oo}; \quad \Sigma_k^{-1,mo} = 0; \quad \Sigma_k^{-1,mm} = \beta \mathbf{I}^{mm}.$$

So,

$$\begin{aligned} d_{kn}^2(\mathbf{t}) &= \begin{pmatrix} \mathbf{t}_n^o - \mathbf{y}_k^o \\ \mathbf{t}_n^m - \mathbf{y}_k^m \end{pmatrix}^{\mathrm{T}} \beta \begin{pmatrix} \mathbf{I}^{oo} & 0 \\ 0 & \mathbf{I}^{mm} \end{pmatrix} \begin{pmatrix} \mathbf{t}_n^o - \mathrm{y}_k^o \\ \mathbf{t}_n^m - \mathbf{y}_k^m \end{pmatrix} \\ &= \beta \begin{pmatrix} \mathbf{t}_n^o - \mathbf{y}_k^o \\ \mathbf{t}_n^m - \mathbf{y}_k^m \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} \mathbf{t}_n^o - \mathbf{y}_k^o \\ \mathbf{t}_n^m - \mathbf{y}_k^m \end{pmatrix} \\ &= \beta[\| \mathbf{t}_n^o - \mathbf{y}_k^o \|^2 + \| \mathbf{t}_n^m - \mathbf{y}_k^m) \|^2], \end{aligned}$$

and it follows from equation A.3 that

$$p(\mathbf{t}_n^o, \mathbf{t}_n^m | \mathbf{x}_k, \mathbf{W}, \beta) = \left(\frac{\beta}{2\pi}\right)^{D/2} \exp\Big\{ -\frac{\beta}{2}\Big[\parallel \mathbf{t}_n^o - \mathbf{y}_k^o \parallel^2 + \parallel \mathbf{t}_n^m - \mathbf{y}_k^m) \parallel^2 \Big]\Big\}.$$

Using the binary indicator variables $\mathcal{Z} = \{\mathbf{z}_n\}_{n=1}^N$, defined such that $\mathbf{z}_n = (z_{1n}, \ldots, z_{kn})$ and $z_{kn} = 1$ if $\mathbf{t}_n$ is generated by Gaussian k, a complete-data negative log likelihood function can be written:

$$E_{comp} = \sum_n^N \sum_k^K z_{kn}\Big\{ \frac{D}{2}\ln(2\pi) - \frac{D}{2}\ln\beta + \frac{\beta}{2}\Big[\parallel \mathbf{t}_n^o - \mathbf{y}_k^o \parallel^2 + \parallel \mathbf{t}_n^m - \mathbf{y}_k^m) \parallel^2 \Big]\Big\}$$

For this model when there is no missing data, the $E-$step computes $\langle z_{kn} | \mathbf{t}_n, \mathbf{W}, \beta \rangle$, which we denote by posterior R_{kn}. This is the probability that Gaussian k generated data point n.

$$R_{kn} = \frac{p_{kn}}{\sum_{k'}^K p_{k'n}} \tag{A.4}$$

Let

$$\Delta = z_{kn}\Big\{ \frac{D}{2}\ln(2\pi) - \frac{D}{2}\ln\beta + \frac{\beta}{2}\Big[\parallel \mathbf{t}_n^o - \mathbf{y}_k^o \parallel^2 + \parallel \mathbf{t}_n^m - \mathbf{y}_k^m) \parallel^2 \Big]\Big\} \tag{A.5}$$

We now consider the expectation of the complete-data negative log likelihood in the form:

$$\begin{aligned}\langle \Delta \rangle &= \langle z_{kn} \rangle \Big(\frac{D}{2}\ln(2\pi) - \frac{D}{2}\ln\beta \Big) + \langle z_{kn} \frac{\beta}{2} \parallel \mathbf{t}_n^o - \mathbf{y}_k^o \parallel^2 \rangle \\ &\quad + \langle \parallel \mathbf{t}_n^m - \mathbf{y}_k^m) \parallel^2 \rangle\end{aligned}$$

For the first and second terms on the right hand side, we note that only the indicator variables z_{kn} are missing, and so the expectation is $\langle z_{kn} | \mathbf{t}_n^o, \theta \rangle = R_{kn}$ (Ghahramani and Jordan, 1994), the responsibility as defined in (A.4) measured only on the observed dimensions of $\mathbf{t}_n$.

As for the third term in equation (A.5), both z_{kn} and $\mathbf{t}_n^m$ are missing. Considering the **sufficient statistics** for the parameters,

$$\begin{aligned}\text{Term3} &= \frac{\beta}{2}\langle z_{kn} | \mathbf{t}_n^o, \theta \rangle \mathbf{tr}\Big(\langle (\mathbf{t}_n^m - \mathbf{y}_k^m)(\mathbf{t}_n^m - \mathbf{y}_k^m)^{\mathrm{T}} | z_{kn} = 1, \mathbf{t}_n^o, \theta \rangle \Big) \\ &= \frac{\beta}{2}\langle z_{kn} | \mathbf{t}_n^o, \theta \rangle \mathbf{tr}\Big(\langle \mathbf{t}_n^m \mathbf{t}_n^{m^T} | z_{kn} = 1, \mathbf{t}_n^o, \theta \rangle - \\ &\quad 2\langle \mathbf{t}_n^m | z_{kn} = 1, \mathbf{t}_n^o, \theta \rangle \mathbf{y}_k^{m^T} + \mathbf{y}_k^m \mathbf{y}_k^{m^T} \Big).\end{aligned} \tag{A.6}$$

For the GTM,

$$\langle z_{kn} \mathbf{t}_n^m | \mathbf{t}_n^o, \theta \rangle = \langle z_{kn} | \mathbf{t}_n^o, \theta \rangle \langle \mathbf{t}_n^m | z_{kn} = 1, \mathbf{t}_n^o, \theta \rangle = R_{kn} \hat{\mathbf{t}}_{kn}^m = R_{kn} (\mu_k^m)^{old} = R_{kn} (\mathbf{y}_k^m)^{old} \tag{A.7}$$

$$\begin{aligned}\langle z_{kn} \mathbf{t}_n^m \mathbf{t}_n^{m^T} | \mathbf{t}_n^o, \theta \rangle &= R_{kn} (\Sigma_k^{mm} - \Sigma_k^{mo} \Sigma_j^{oo,-1} \Sigma_k^{mo^T} + \hat{\mathbf{t}}_k^m \hat{\mathbf{t}}_k^{m^T} | z_{kn} = 1) \\ &= R_{kn} [(\Sigma_k^{mm})^{old} + (\mathbf{y}_k^m)^{old} (\mathbf{y}_k^{m^T})^{old}]\end{aligned} \tag{A.8}$$

So, for the third term (A.6), we have:

$$\text{Term3} \quad = \quad \frac{\beta}{2} R_{kn} \mathbf{tr}\Big[(\beta^{-1})^{old}\mathbf{I}^{mm} + \Big((\mathbf{y}_k^m)^{old} - (\mathbf{y}_k^m)\Big)\Big((\mathbf{y}_k^m)^{old} - (\mathbf{y}_k^m)\Big)^T\Big].$$

For the GTM:

$$\mathbf{y}_k(\mathbf{x};\mathbf{W}) = \mathbf{W}\boldsymbol{\phi}(\mathbf{x}_k)$$

$$\begin{aligned}\frac{\partial(\sum_{n,k}^{N,K}\text{term2})}{\partial \mathbf{W}} &= \frac{\sum_{n,k}^{N,K}\partial\{R_{kn}\frac{\beta}{2}\parallel \mathbf{t}_n^o - [\mathbf{W}\boldsymbol{\phi}(\mathbf{x}_k)]_k^o \parallel^2\}}{\partial \mathbf{W}} \\ &= -\sum_{n,k}^{N,K} R_{kn}\boldsymbol{\phi}^{\mathrm{T}}(\mathbf{x}_k)\beta\left(\mathbf{t}_n^o - [\mathbf{W}\boldsymbol{\phi}(\mathbf{x}_k)]_k^o\right),\end{aligned}$$

$$\begin{aligned}\frac{\partial(\sum_{n,k}^{N,K}\text{term3})}{\partial \mathbf{W}} &= \frac{\sum_{n,k}^{N,K}\partial\Big[R_{kn}\frac{\beta}{2}\mathbf{tr}\left([\mathbf{W}\boldsymbol{\phi}(\mathbf{x}_k)]_k^m[\mathbf{W}\boldsymbol{\phi}(\mathbf{x}_k)]_k^{m^T} - 2[\mathbf{W}\boldsymbol{\phi}(\mathbf{x}_k)]_k^{m^T}(\mathbf{y}_k^m)^{old}\right)\Big]}{\partial \mathbf{W}} \\ &= -\sum_{n,k}^{N,K} R_{kn}\boldsymbol{\phi}^{\mathrm{T}}(\mathbf{x}_k)\beta\left(\hat{\mathbf{t}}_{kn}^m - [\mathbf{W}\boldsymbol{\phi}(\mathbf{x}_k)]_k^m\right),\end{aligned}$$

where $\hat{\mathbf{t}}_{kn}^m = \mathrm{E}[\mathbf{t}_n^m | z_{kn} = 1, \mathbf{t}_n^o, \theta] = \mathbf{y}_k^{m^{old}}$ substitute for missing values.

setting

$$\frac{\partial(\mathrm{E}(l))}{\partial \mathbf{W}} = 0;$$

For observed data and filled-in data, we have

$$-\sum_{n,k}^{N,K} R_{kn}\boldsymbol{\phi}^{\mathrm{T}}(\mathbf{x}_k)\beta\left(\mathbf{t}_n^o - [\mathbf{W}\boldsymbol{\phi}(\mathbf{x}_k)]_k^o\right) - \sum_{n,k}^{N,K} R_{kn}\boldsymbol{\phi}^{\mathrm{T}}(\mathbf{x}_k)\beta\left(\hat{\mathbf{t}}_{kn}^m - [\mathbf{W}\boldsymbol{\phi}(\mathbf{x}_k)]_k^m\right) = 0.$$

Now we write both observed data $\mathbf{t}_n^o$ and missing values filled by $\hat{\mathbf{t}}_{kn}^m$ into vector $\mathbf{t}_n$,

$$\sum_{n,k}^{N,K} R_{kn}\boldsymbol{\phi}^{\mathrm{T}}(\mathbf{x}_k)\beta\mathbf{W}\boldsymbol{\phi}(\mathbf{x}_k) = \sum_{n,k}^{N,K} R_{kn}\boldsymbol{\phi}^{\mathrm{T}}(\mathbf{x}_k)\beta\mathbf{t}_n.$$

This can be re-written as matrix form

$$\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{G}\boldsymbol{\Phi}\mathbf{W} = \boldsymbol{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{T}.$$

This concludes the $M-$step for the weights, we now consider the $M-$step for the variance:

$$\frac{\partial(\mathrm{E}(l))}{\partial \beta} \quad = \quad \frac{\sum_{n,k}^{N,K}\partial\{-\mathrm{E}[z_{kn}]\frac{D}{2}\ln\beta + R_{kn}\frac{\beta}{2}\parallel \mathbf{t}_n^o - \mathbf{y}_k^o \parallel^2 + \text{term3}\}}{\partial \beta}$$

$$\sum_{n,k}^{N,K}\frac{\partial(-R_{kn}\frac{D}{2}\ln\beta)}{\partial\beta} = \sum_{n,k}^{N,K} -R_{kn}\frac{D}{2}\beta^{-1} = -\frac{ND}{2}\beta^{-1}$$

$$\frac{\partial(\text{term3})}{\partial\beta} = \sum_{n,k}^{N,K}\frac{1}{2}R_{kn}\Big[n_m(\beta^{-1})^{old} + (\mathbf{y}_k^m)^{old^T}(\mathbf{y}_k^m)^{old} - 2(\mathbf{y}_k^m)^{old^T}\mathbf{y}_k^m + \mathbf{y}_k^{m^T}\mathbf{y}_k^m\Big]$$

Thus

$$\frac{\partial(\mathrm{E}(l))}{\partial\beta} = -\frac{ND}{2}\beta^{-1} + \frac{1}{2}\sum_{n,k}^{N,K} R_{kn} \parallel \mathbf{t}_n^o - \mathbf{y}_k^o \parallel^2 + \frac{1}{2}\sum_{n,k}^{N,K} R_{kn}\Big[n_m(\beta^{-1})^{old} + ||(\mathbf{y}_k^m)^{old} - \mathbf{y}_k^m||^2\Big]$$

Setting

$$\frac{\partial(\mathrm{E}(l))}{\partial\beta} = 0,$$

we get the update formula

$$\beta^{-1} = \frac{1}{ND}\sum_{n,k}^{N,K} R_{kn}\Big(\|\mathbf{t}_n^o - \mathbf{y}_k^o\|^2 + \langle\|\mathbf{t}_n^m - \mathbf{y}_k^m\|^2 | z_{kn} = 1\rangle\Big),$$

where

$$\langle\|\mathbf{t}_n^m - \mathbf{y}_k^m\|^2 | z_{kn} = 1\rangle = n_m(\beta^{-1})^{old} + ||(\mathbf{y}_k^m)^{old} - \mathbf{y}_k^m||^2,$$

and n_m is number of missing values in data point $\mathbf{t}_n$.

Appendix B

Quantities for computing magnification factors

The exact form of the matrices $\mathcal{I}$ is dependent on the specific noise-model being employed. There require the first derivatives of the inverse link function $\mathbf{g}(\cdot)$. We will provide here the expressions for the models utilized in the experiments reported herein.

B.1 Independent Gaussian noise model

The Gaussian model the only member of the exponential family of distributions which is characterised by a quadratic cumulant function

$$\mathcal{G}_d(\mathbf{y}) = \frac{1}{2} y_d^2. \tag{B.1}$$

Therefore, it has a linear inverse-link function and higher derivatives vanish.

$$\mathbf{g}_{d'}(\mathbf{y}) = y_{d'}, \tag{B.2}$$

$$\frac{\partial \mathbf{g}_{d'}(\mathbf{y})}{\partial y_d} = 0. \tag{B.3}$$

B.2 Independent binomial noise model

In the case of binomial model, the cumulant function is

$$\mathcal{G}_d(\mathbf{y}) = \log(1 + \exp(y_d)). \tag{B.4}$$

The required derivative are then computed as follows:

$$\mathbf{g}_{d'}(\mathbf{y}) = \frac{\exp(y_{d'})}{1 + \exp(y_{d'})}, \tag{B.5}$$

$$\frac{\partial \mathbf{g}_{d'}(\mathbf{y})}{\partial y_d} = \begin{cases} 0 & d \neq d' \\ \mathbf{g}_d(\mathbf{y})(1 - \mathbf{g}_d(\mathbf{y})) & d = d'. \end{cases} \tag{B.6}$$

It can be seen that for independent noise models, the Fisher information matrix $\mathcal{I}$ is diagonal.

B.3 Multinomial noise model

The multinomial member of the exponential family of distributions is identified by the cumulant function of the following form,

$$\mathcal{G}(\mathbf{y}) = \log(\sum_{d=1:D} \exp(y_d)). \tag{B.7}$$

Accordingly, the first derivative is given by

$$\mathrm{g}_{d'}(\mathbf{y}) = \frac{\exp(y_{d'})}{\sum_{d''=1}^{D} \exp(y_{d''})}, \tag{B.8}$$

$$\frac{\partial \mathrm{g}_{d'}(\mathbf{y})}{\partial y_d} = \begin{cases} -\mathrm{g}_{d'}(\mathbf{y})\mathrm{g}_d(\mathbf{y}) & d \neq d' \\ \mathrm{g}_{d'}(\mathbf{y}) - \mathrm{g}_{d'}(\mathbf{y})\mathrm{g}_d(\mathbf{y}) & d = d'. \end{cases} \tag{B.9}$$

Appendix C

Mixing Coefficients

In the hierarchical latent trait model algorithm, maximising (4.20) with respect to the parent-conditional mixture coefficients $P(\mathcal{M}|\mathcal{N})$ in the $M-$step of the EM must take account of the constraint

$$\sum_{\mathcal{M}\in Children(\mathcal{N})} P(\mathcal{M}|\mathcal{N}) = 1. \tag{C.1}$$

This can be achieved by introducing a Lagrange multiplier $\lambda_{\mathcal{N}}$ and maximising

$$\sum_{n=1}^{N} \sum_{\mathcal{N}\in Nodes(\ell)} P(\mathcal{N}|\mathbf{t}_n) \sum_{\mathcal{M}\in Children(\mathcal{N})} P(\mathcal{M}|\mathcal{N},\mathbf{t}_n) \ln P(\mathcal{M}|\mathcal{N}) + \lambda_{\mathcal{N}} \left(\sum_{\mathcal{M}\in Children(\mathcal{N})} P(\mathcal{M}|\mathcal{N}) - 1 \right). \tag{C.2}$$

For a specific $\mathcal{N} = Parent(\mathcal{M})$, we have

$$C = \sum_{n=1}^{N} P(\mathcal{N}|\mathbf{t}_n) \sum_{\mathcal{M}'\in Children(\mathcal{N})} P(\mathcal{M}'|\mathcal{N},\mathbf{t}_n) \ln P(\mathcal{M}'|\mathcal{N}) + \lambda_{\mathcal{N}} \left(\sum_{\mathcal{M}'\in Children(\mathcal{N})} P(\mathcal{M}'|\mathcal{N}) - 1 \right). \tag{C.3}$$

We differentiate C with respect to $P(\mathcal{M}|\mathcal{N})$ and set it zero,

$$\frac{\partial C}{\partial P(\mathcal{M}|\mathcal{N})} = \sum_{n=1}^{N} P(\mathcal{N}|\mathbf{t}_n) P(\mathcal{M}|\mathcal{N},\mathbf{t}_n) \frac{1}{P(\mathcal{M}|\mathcal{N})} + \lambda_{\mathcal{N}} = 0. \tag{C.4}$$

We have

$$\sum_{n=1}^{N} P(\mathcal{N}|\mathbf{t}_n) P(\mathcal{M}|\mathcal{N},\mathbf{t}_n) + \lambda_{\mathcal{N}} P(\mathcal{M}|\mathcal{N}) = 0, \tag{C.5}$$

$$P(\mathcal{M}|\mathcal{N}) = -\frac{\sum_{n=1}^{N} P(\mathcal{N}|\mathbf{t}_n) P(\mathcal{M}|\mathcal{N},\mathbf{t}_n)}{\lambda_{\mathcal{N}}}. \tag{C.6}$$

Substituting (C.6) into (C.1), we have

$$-\frac{1}{\lambda_{\mathcal{N}}} \sum_{\mathcal{M}'\in Children(\mathcal{N})} \sum_{n=1}^{N} P(\mathcal{N}|\mathbf{t}_n) P(\mathcal{M}'|\mathcal{N},\mathbf{t}_n) = 1 \tag{C.7}$$

Furthermore, we obtain

$$\begin{aligned} \lambda_{\mathcal{N}} &= -\sum_{n=1}^{N} P\left(\mathcal{N}|\mathbf{t}_n\right) \sum_{\mathcal{M}' \in Children(\mathcal{N})} P(\mathcal{M}'|\mathcal{N}, \mathbf{t}_n) \\ &= -\sum_{n=1}^{N} P\left(\mathcal{N}|\mathbf{t}_n\right). \end{aligned} \tag{C.8}$$

Finally taking (C.8) back into (C.6), we have

$$P(\mathcal{M}|\mathcal{N}) = \frac{\sum_{n=1}^{N} P\left(\mathcal{N}|\mathbf{t}_n\right) P(\mathcal{M}|\mathcal{N}, \mathbf{t}_n)}{\sum_{n=1}^{N} P\left(\mathcal{N}|\mathbf{t}_n\right)}. \tag{C.9}$$

Appendix D

Derivation of minimum message length

D.1 MML in a univariate case

Let us consider the scalar parameter case first. We follow (Lanterman, 2001) to reformulate the derivation of the MML criterion using our notation. Let θ_{tr} be a finite precision version of θ with quantization length s. The prior probability that $\theta \in [\theta_{tr} - \frac{s}{2}, \theta_{tr} + \frac{s}{2},]$ is approximately $sp(\theta_{tr})$. Expanding the log-likelihood as a function of the truncated value in a second order Taylor series around θ yields

$$-\log p(\zeta|\theta_{tr}) \approx -\log p(\zeta|\theta) - (\theta_{tr} - \theta)\frac{\partial}{\partial\theta}\log p(\zeta|\theta) - \frac{1}{2}(\theta_{tr} - \theta)^2\frac{\partial^2}{\partial\theta^2}\log p(\zeta|\theta). \tag{D.1}$$

We assume that the quantization error is uniformly distributed in $[-\frac{s}{2}, \frac{s}{2}]$, and so $\langle\theta - \theta_{tr}\rangle = 0$; $\langle(\theta - \theta_{tr})^2\rangle = \frac{s^2}{12}$; and we approximate $sp(\theta_{tr}) \approx sp(\theta)$ assuming the prior is smooth enough.

Now the expected value of the message length (see equation (5.18)) is given by

$$\langle\text{Length}(\theta_{tr}, \zeta)\rangle \approx -\log s - \log p(\theta) - \log p(\zeta|\theta) + \frac{s^2}{24}\mathcal{I}(\zeta, \theta), \tag{D.2}$$

where the first order Taylor series term of the right hand side in equation (D.1) vanishes after taking the expectation due to assumptions on the uniform distribution of the quantization error, and $\mathcal{I}(\zeta, \theta)$ is the *observed* Fisher information given by $\mathcal{I}(\zeta, \theta) \equiv -\frac{\partial^2 \log p(\zeta|\theta)}{\partial\theta^2}$.

Setting the derivative of (D.2) with respect to s to zero, we have

$$s = \sqrt{\frac{12}{\mathcal{I}(\zeta, \theta)}} \quad . \tag{D.3}$$

Substituting (D.3) into (D.2), it yields the following equation

$$\langle\text{Length}(\theta_{tr}, \zeta)\rangle \approx -\log p(\theta) - \log p(\zeta|\theta) + \frac{1}{2}\log\mathcal{I}(\zeta, \theta) + \frac{1}{2}(1 - \log 12). \tag{D.4}$$

Approximating the *observed* Fisher information by the *expected* Fisher information, which has the form

$$\mathrm{I}(\theta) \quad = \quad -\langle \frac{\partial^2 \log p(\zeta|\theta)}{\partial \theta^2} \rangle, \tag{D.5}$$

yields the approximation

$$\langle \mathrm{Length}(\theta_{tr}, \zeta) \rangle \approx -\log p(\theta) - \log p(\zeta|\theta) + \frac{1}{2} \log \mathrm{I}(\theta) + \frac{1}{2}(1 - \log 12). \tag{D.6}$$

The θ which minimises (D.6) is the MML estimate.

D.2 MML in multiple dimensions

Now consider the multivariate case. This section follows (Oliver and Baxter, 1995). We assume that the c dimensional parameter space is partitioned into regions of volume V which is a function of $\boldsymbol{\theta}$. So the prior probability is approximately $Vp(\boldsymbol{\theta})$. In this case, the expected value of the message length is

$$\langle \mathrm{Length}(\boldsymbol{\theta}_{tr}, \zeta) \rangle \approx -\log (Vp(\boldsymbol{\theta})) - \log p(\zeta|\boldsymbol{\theta}) - \frac{1}{2}\frac{1}{V} \int_V (\boldsymbol{\theta} - \boldsymbol{\theta}_{tr})^T \frac{\partial^2 \log p(\zeta|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^2} (\boldsymbol{\theta} - \boldsymbol{\theta}_{tr}) dv. \tag{D.7}$$

To make the message decodable, we make the approximation of replacing $-\frac{\partial^2 \log p(\zeta|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^2}$ with the *expected* Fisher information matrix $\mathbf{I}(\boldsymbol{\theta})$ and denote $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta} - \boldsymbol{\theta}_{tr}$, so the integral part of (D.7), denoted by I, is approximated by

$$I \approx \frac{1}{2}\frac{1}{V} \int_V \hat{\boldsymbol{\theta}}^T \mathbf{I}(\boldsymbol{\theta}) \hat{\boldsymbol{\theta}} dv.$$

It will be convenient to make a change in coordinates $\hat{\boldsymbol{\vartheta}} = B^{-1}\hat{\boldsymbol{\theta}}$, where B is chosen so that $\hat{\boldsymbol{\theta}}^T \mathbf{I}(\boldsymbol{\theta}) \hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\vartheta}}^T \hat{\boldsymbol{\vartheta}}$.

Let $p(\boldsymbol{\vartheta})$ be the transformed prior density, and let volume V in the $\boldsymbol{\theta}$-space map into volume U in the $\boldsymbol{\vartheta}$-space. The expected message length is given by

$$\begin{aligned} \langle \mathrm{Length}(\boldsymbol{\theta}_{tr}, \zeta) \rangle \quad &\approx \quad -\log(Up(\boldsymbol{\vartheta})) - \log p(\zeta|\boldsymbol{\theta}) + \frac{1}{2}\frac{1}{U} \int_U (\hat{\boldsymbol{\vartheta}}^T \hat{\boldsymbol{\vartheta}}) du \\ &= \quad -\log U - \log p(\boldsymbol{\vartheta}) - \log p(\zeta|\theta) + \frac{1}{2} \langle \hat{\boldsymbol{\vartheta}}^T \hat{\boldsymbol{\vartheta}} \rangle. \end{aligned} \tag{D.8}$$

Wallace and Freeman consider quantizing in multiple dimensions using optimal quantizing lattices. For example, in two dimensions, the optimal quantizing lattice forms a hexagonal grid. In three dimensions, the optimal lattice is a body-centres cubic lattice. In higher dimensions, the optimal quantizing lattices are actually unknown. We use κ_c to denote a constant relating to the geometry of the c-dimensional lattice. The expected value of $\hat{\boldsymbol{\vartheta}}^T \hat{\boldsymbol{\vartheta}}$ can be expressed in terms of the c-dimensional optimal quantizing lattice constant κ_c,

$$\langle \hat{\boldsymbol{\vartheta}}^T \hat{\boldsymbol{\vartheta}} \rangle = c\kappa_c U^{2/c}, \tag{D.9}$$

where $\kappa_1 = 1/12$ since $(\int_{-1/2}^{1/2} \boldsymbol{\vartheta}^2 d\boldsymbol{\vartheta} = 1/12)$. Conway and Sloane (1993) give bounds on the c-dimensional optional quantizing lattice constants, κ_c.

Therefore,

$$\langle \mathrm{Length}(\boldsymbol{\theta}_{tr}, \zeta) \rangle \approx -\log U - \log p(\boldsymbol{\vartheta}) - \log p(\zeta|\theta) + \frac{c}{2} \kappa_c U^{2/c}. \tag{D.10}$$

We differentiate equation (D.10) with respect to U and set it to zero. Then we have

$$U = \kappa_c^{-c/2}.$$

In these new coordinates, the prior $P(\vartheta)$ is

$$p(\vartheta) = p(\boldsymbol{\theta})\frac{dV}{dU} = p(\boldsymbol{\theta})\frac{1}{\text{Jacob}(B^{-1})} = \frac{p(\boldsymbol{\theta})}{\sqrt{|\mathbf{I}(\boldsymbol{\theta})|}}. \tag{D.11}$$

For more details, see (Oliver and Baxter, 1995) (section 5, Appendix 1).

Using the above equation to translate back into the $\boldsymbol{\theta}$-space, the message length is

$$\langle \text{Length}(\boldsymbol{\theta}_{tr}, \zeta) \rangle = -\log p(\boldsymbol{\theta}) - \log p(\zeta|\boldsymbol{\theta}) + \frac{1}{2}\log|\mathbf{I}(\boldsymbol{\theta})| + \frac{c}{2}(1 + \log \kappa_c). \tag{D.12}$$

The MML estimate of $\boldsymbol{\theta}$ is the $\boldsymbol{\theta}$ which minimises (D.12).

Appendix E

The MML Framework for Mixture Models

E.1 The cost function of MML for mixture models

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{ -\log p(\boldsymbol{\theta}) - \log p(\zeta|\boldsymbol{\theta}) + \frac{1}{2}\log|\mathbf{I}(\boldsymbol{\theta})| + \frac{c}{2}(1+\log\frac{1}{12}) \right\} \tag{E.1}$$

$$\begin{aligned}
\jmath(\boldsymbol{\theta},\zeta) &= -\log\left[p(P(1),\ldots,P(A))\prod_{a=1}^{A} p(\boldsymbol{\theta}_a)\right] + \frac{1}{2}\log|\mathbf{I}_c(\boldsymbol{\theta})| + \frac{(Q+1)A}{2}(1-\log 12) - \log p(\zeta|\boldsymbol{\theta}) \\
&\approx -\log(P(1),\ldots,P(A))^{-\frac{1}{2}} - \sum_{a=1}^{A}\log|\mathbf{I}^{(1)}(\boldsymbol{\theta}_a)|^{\frac{1}{2}} + \frac{1}{2}\left[(Q+1)A\log N + (Q-1)\sum_{a=1}^{A}\log P(a)\right. \\
&\quad \left. + \sum_{a=1}^{A}\log|\mathbf{I}^{(1)}(\boldsymbol{\theta}_a)| + \frac{(Q+1)A}{2}(1-\log 12) - \log p(\zeta|\boldsymbol{\theta})\right] \qquad \text{(see section 5.3.1)} \\
&= \frac{1}{2}\sum_{a=1}^{A}\log P(a) - \frac{1}{2}\sum_{a=1}^{A}\log|\mathbf{I}^{(1)}(\boldsymbol{\theta}_a)| + \frac{1}{2}(Q+1)A\log N + \frac{Q-1}{2}\sum_{a=1}^{A}\log P(a) \\
&\quad + \frac{1}{2}\sum_{a=1}^{A}\log|\mathbf{I}^{(1)}(\boldsymbol{\theta}_a)| + \frac{(Q+1)A}{2}(1-\log 12) - \log p(\zeta|\boldsymbol{\theta}) \\
&= \frac{Q}{2}\sum_{a=1}^{A}\log P(a) + \frac{1}{2}(Q+1)A\log N + \frac{(Q+1)A}{2} + \frac{(Q+1)A}{2}\log\frac{1}{12} - \log p(\zeta|\boldsymbol{\theta}) \\
&= \frac{Q}{2}\sum_{a=1}^{A}\log P(a) + \frac{(Q+1)A}{2}\log\frac{N}{12} + \frac{(Q+1)A}{2} - \log p(\zeta|\boldsymbol{\theta}) \\
&= \frac{Q}{2}\sum_{a=1}^{A}\log P(a) + \frac{QA}{2}\log\frac{N}{12} + \frac{A}{2}\log\frac{N}{12} + \frac{(Q+1)A}{2} - \log p(\zeta|\boldsymbol{\theta}) \\
&= \frac{Q}{2}\sum_{a=1}^{A}\log P(a) + \frac{Q}{2}\sum_{a=1}^{A}\log\frac{N}{12} + \frac{A}{2}\log\frac{N}{12} + \frac{(Q+1)A}{2} - \log p(\zeta|\boldsymbol{\theta}) \\
&= \frac{Q}{2}\sum_{a=1}^{A}\log\frac{NP(a)}{12} + \frac{A}{2}\log\frac{N}{12} + \frac{(Q+1)A}{2} - \log p(\zeta|\boldsymbol{\theta})
\end{aligned} \tag{E.2}$$

Since we only code the parameters of mixture components with positive prior, we have

$$\jmath(\boldsymbol{\theta}, \zeta) = \frac{Q}{2} \sum_{a:P(a)>0} \log \frac{NP(a)}{12} + \frac{A_+}{2} \log \frac{N}{12} + \frac{(Q+1)A_+}{2} - \log p(\zeta|\boldsymbol{\theta}), \tag{E.3}$$

where A_+ denotes the number of components whose priors are positive.

E.2 Priors

The cost function of MML for mixture models is given by

$$\jmath(\boldsymbol{\theta}, \zeta) = \frac{Q}{2} \sum_{a:P(a)>0} \log \frac{NP(a)}{12} + \frac{A_+}{2} \log \frac{N}{12} + \frac{(Q+1)A_+}{2} - \log p(\zeta|\boldsymbol{\theta}), \tag{E.4}$$

where $-\log p(\zeta|\boldsymbol{\theta})$ is the negative log likelihood E having form as follows:

$$E = \sum_{n=1}^{N} \log \left\{ \sum_{a=1}^{A} P(a) p(\mathbf{t}_n|\boldsymbol{\theta}_a) \right\}. \tag{E.5}$$

Introducing assignment variables z_{an}, we have

$$E_{comp} = -\sum_{n=1}^{N} \sum_{a=1}^{A} z_{an} \log \{P(a) p(\mathbf{t}_n|\boldsymbol{\theta}_a)\}. \tag{E.6}$$

The expectation of cost function of MML for mixture models is given by

$$\begin{aligned} \langle \jmath(\boldsymbol{\theta}, \zeta) \rangle = & \frac{Q}{2} \sum_{a=1}^{A} \log \frac{NP(a)}{12} + \frac{A_+}{2} \log \frac{N}{12} + \frac{(Q+1)A_+}{2} \\ & - \sum_{a=1}^{A} \log P(a) \sum_{n=1}^{N} P(a|\mathbf{t}_n) - \sum_{a=1}^{A} \sum_{n=1}^{N} P(a|\mathbf{t}_n) \log p(\mathbf{t}_n|\boldsymbol{\theta}_a). \end{aligned} \tag{E.7}$$

Minimisation (E.7) with respect to $P(a)$ must take account of the constraint

$$\sum_{a=1}^{A} P(a) = 1, \quad \text{and} \quad P(a) \geq 0, \quad a = 1, 2, \ldots, A.$$

This can be achieved by introducing a Lagrange multiplier λ and minimising CF which is given by

$$\mathrm{CF} = \frac{Q}{2} \sum_{a'=1}^{A} \log P(a') - \sum_{a'=1}^{A} \log P(a') \sum_{n=1}^{N} P(a'|\mathbf{t}_n) + \lambda \left(\sum_{a'=1}^{A} P(a') - 1 \right). \tag{E.8}$$

To differentiate CF with respect to $P(a)$ and set it zero, we have

$$\frac{\partial \mathrm{CF}}{\partial P(a)} = \frac{1}{P(a)} \left[\frac{Q}{2} - \sum_{n=1}^{N} P(a|\mathbf{t}_n) \right] + \lambda = 0. \tag{E.9}$$

Then we obtain

$$P(a) = \frac{\sum_{n=1}^{N} P(a|\mathbf{t}_n) - \frac{Q}{2}}{\lambda}. \tag{E.10}$$

To obtain the value for λ, we use the constraint

$$\sum_{a'=1}^{A} \frac{\sum_{n=1}^{N} P(a'|\mathbf{t}_n) - \frac{Q}{2}}{\lambda} - 1 = 0. \tag{E.11}$$

Then we have

$$\lambda = \sum_{a'=1}^{A} \left(\sum_{n=1}^{N} P(a'|\mathbf{t}_n) - \frac{Q}{2} \right). \tag{E.12}$$

Taking (E.12) back to (E.10), we have

$$P(a) = \frac{\sum_{n=1}^{N} P(a|\mathbf{t}_n) - \frac{Q}{2}}{\sum_{a'=1}^{A} \left(\sum_{n=1}^{N} P(a'|\mathbf{t}_n) - \frac{Q}{2} \right)}. \tag{E.13}$$

However, since $P(a) \geq 0$ for all $a = 1, 2, \ldots, A$, we have

$$P(a) = \frac{\max\left\{0, \sum_{n=1}^{N} P(a|\mathbf{t}_n) - \frac{Q}{2}\right\}}{\sum_{a'=1}^{A} \max\left\{0, \sum_{n=1}^{N} P(a'|\mathbf{t}_n) - \frac{Q}{2}\right\}}. \tag{E.14}$$

References

Akaike, H. 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control* **19**, 716–723.

Amari, S. 1985. *Differential Geometrical Methods in Statistics*. Springer Verlag.

Aurenhammer, F. 1991. Voronoi diagrams-survey of a fundamental geometric data structure. *ACM Computing Surveys* **3**, 345–405.

Barndorff-Nielsen, O. 1978. *Information and Exponential Families in Statistical Theory*. Wiley, Chichester.

Bartholomew, D. and M. Knott 1999. *Latent variable models and factor analysis* (2nd ed.), Volume 7 of *Kendall's library of statistics*. London, Arnold.

Bates, D. and D. Watts 1980. Relative curvature measures of nonlinearity with discussion. *J. R. Stat. Soc. B* **42**, 1–25.

Bernardo, J. and A. Smith 1994. *Bayesian Theory*. Chichester, UK: J. Wiley & Sons.

Bishop, C. 1995. *Neural Networks for Pattern Recognition*. New York, N.Y.: Oxford University Press.

Bishop, C., G. Hinton, and I. Strachan 1997. GTM through time. In *Proceedings IEE Fifth International Conference on Artificial Neural Networks*, pp. pp 111–116. London., IEE.

Bishop, C., M. Svensén, and C. Williams 1997. GTM: a principled alternative to the Self-Organizing Map. In *Advances in Neural Information Processing System*, Volume 9. MIT Press.

Bishop, C., M. Svensén, and C. Williams 1997. Magnification factors for the SOM and GTM algorithms. In *Proceedings 1997 Workshop on Self-Organizing Maps*. Helsinki, Finland.

Bishop, C., M. Svensén, and C. Williams 1998. Developments of the generative topographic mapping. *Neurocomputing* **21**, 203–224.

Bishop, C., M. Svensén, and C. Williams 1998. GTM: the generative topographic mapping. *Neural Computation* **10** (1), 215–235.

Bishop, C. and M. Tipping 1998. A hierarchical latent variable model for data visualisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20** (3), 281–293.

Bozdogan, H. 1983, June. Determining the number of component clusters in the standard multivariate normal mixture model using model-selection criteria. Technical Report TR UIC/DQM/A83-1, Quantitative Methods Dept., University of Illinois, Chicago, Illinois 60680.

Bozdogan, H. 1990. On the information-based measure of covariance complexity and it s applications to the evaluation of multivariate linear models. *Communications in Statistics - Theory and Methods* **19**, 221–278.

Bozdogan, H. 1993. Choosing the number of component clusters in the mixture-model using a new informational complexity criterion of the inverse-Fisher information matrix. In O. Optiz, B. Lausen, and R. Klar (Eds.), *Information and Classification*, pp. 40–54. Heidelberg: Springer-Verlag.

Buck, S. 1960. A method of estimation of missing values in multivariate data suitable for use with an electronic computer. *J. Roy. Statist. Soci* **B22**, 302–306.

Cassie, R. 1954. Some uses of probability paper in the analysis of size frequency distributions. *Australian Journal of Marine and Freshwater Research* **5**, 513–522.

Celeux, G., F. F. S. Chrétien, and A. Mkhadri 2001. A component-wise EM algorithm for mixtures. *J. Comput. Graphical Statistics* **10**, 699–712.

Celeux, G. and G. Soromenho 1996. An entropy criterion for assessing the number of clusters in a mixture model. *Classification Journal* **13**, 105–112.

Chrétien, S. and A. Hero 2000. Kullback proximal algorithms for maximum likelihood estimation. *IEEE Trans. on Information Theory* **46**, 1800 1810.

Conway, J. and N. Sloane 1993. *Sphere Packings, Lattices and Groups*. New York: Springer-Verlag. second edition.

Cover, T. and J. Thomas 1991. *Elements of Information Theory*. John Wiley & Sons, Inc.

Dacunha-Castelle, D. and E. Gassiat 1997. The estimation of the order of a mixture model. *Bernoulli* **3**, 279–299.

Dalal, S. and W. Hall 1983. Approximating priors by mixtures of natural conjugate priors. *Jour. of the Royal Statistical Society (B)* **45**.

Deerwester, S., S. Dumais, G. Furnas, T. Landauer, and R. Harshman 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science* **41** (6), 391–407.

Dempster, A., N. Laird, and D. Rubin 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Stat. Soc. B* **39**, 1–38.

Dixon, W. (Ed.) 1983. *BMDP Statistical Software*. Berkeley: University of California Press. 1983 revised printing.

Figueiredo, M. and A. Jain 2002. Unsupervised learning of finite mixture models. *IEEE Transaction on Pattern Analysis and Machine Intelligence* **24**, 381–396.

Friend, S. and R. Stoughton 2002, February. The magic of microarrays. *Scientific American* **286** (2), 34–39.

Ghahramani, Z. and M. I. Jordan 1994. Learning from incomplete data. Technical report, AI Laboratory, MIT.

Green, P. 1998. Penalized likelihood. In *Encyclopedis of Statistical Sciences*, Volume 2. John Wiley & Sons, Inc.

Harding, J. 1948. The use of probability paper for the graphical nalysis of polymodal frequency distributions. *Journal of Marine Biological Association* **28**, 141–153.

Hastie, T. and W. Stuetzle 1989. Principal curves. *Journal of American Statistical Association* **84** (406), 502–516.

Hastie, T. and R. Tibshirani 1996. Discriminant analysis by Gaussian mixtures. *Jour. of the Royal Statistical Society (B)* **58**, 155–176.

Heijne, G. 1986. A new method for predicting signal sequence cleavage sites. *Nucl. Acids Res.* (14), 4683–4690.

Hinton, G., P. Dayan, and M. Revow 1997. Modelling the manifolds of images of handwritten digits. *IEEE Trans. on Neural Networks* **8**, 65–74.

Horn, R. and C. Johnson 1985. *Matrix Analysis*. Cambridge, England: Cambridge University Press.

Jagota, A. 2001. *Microarray Data Analysis and Visualisation*. Bioinformatics By The Bay Press.

Jain, A. K. and R. Dubes 1988. *Algorithms for Clustering Data*. Englewood Cliffs, N. J.: Prentice Hall.

Jain, A. K., R. Duin, and J. Mao 2000. Statistical pattern recognition: a review. *IEEE Trans. on Patt. Anal. and Machine Intell* **22** (1), 4–38.

Jordan, M., Z. Ghahramani, T. Jaakkola, and L. Saul 1998. An introduction to variational methods for graphical. In M. Jordan (Ed.), *Learning in Graphical Model*. Kluwer Academic Publishers.

Kabán, A. and M. Girolami 2001. A combined latent class and trait model for the analysis and visualisation of discrete data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23(8)**, 859–872.

Kaban, A., P. Tino, and M. Girolami 2002. A general framework for a principled hierarchical visualisation of multivariate data. In *International Conference on Intelligent Data Engineering and Automated Learning - IDEAL 2002.*, pp. 17–23. Springer-Verlag.

Kegl, B., A. Krzyzak, T. Linder, and K. Zeger 2000. Learning and design principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22** (3), 281–297.

Kohonen, T. 1995. *Self-Orgainzing Maps*. Berlin: Springer-Verlag.

Kosala, R. and H. Blockeel 2000. Web mining research: a survey. *SIGKDD Explorations* **2**.

Kruskal, J. 1964. Multidimensional scaling by optimising goodness of fit to a nonmetric hypothesis. *Psychometrika* **29** (1), 1–27.

Kullback, S. and R. Leibler 1951. On information and sufficiency. *Annals of Mathematical Statistics* **22**, 79–86.

Lanterman, A. 2001, August. Schwarz, Wallace, and Rissanen: intertwining themes in theories of model order estimation. *International Statistical Review* **69**.

Little, R. 1992, December. Regression with missing X's: a review. *Journal of the American Statistical Association* **87** (420).

Little, R. and D. Rubin 1987. *Statistical Analysis with Missing Data*. John Wiley & Sons, Inc.

Lowe, D. and M. Tipping 1997. Neuroscale: novel topographic feature extraction with radial basis function network. In M. Mozer, M. Jordan, and T. Petsche (Eds.), *Advances in Neural Information Processing System*, Volume 9, pp. 543–549. MIT Press.

MacKay, D. 1995. Probable networks and plausible predictions-a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural System* **6**.

McCullagh, P. and L. Nelder 1985. *Generalized Linear Models*. Chapman and Hall.

Mcgeoch, D. 1985. On the predictive recognition of signal peptide sequences. *Virus Res* **271** (3).

McLachlan, G. and K. Basford 1988. *Mixture Models: Inference and Application to Clustering*. New York: Marcel Dekker.

Mclachlan, G. and T. Krishnan 1997. *The EM Algorithm and Extensions*. Wiley series in probability and statistics. John Wiley & Sons, INC.

McLachlan, G. and D. Peel 2000. *Finite Mixture Models*. New York: John Wiley & Sons.

Minka, T. 2000. Automatic choice of dimensionality for PCA. Technical Report 514, MIT Media Laboratory, Perceptual Computing Section.

Moustaki, I. 1996. A latent trait and a latent class model for mixed observed variables. *British Journal of Mathematical and Statistical Psychology* **49**, 313–334.

Nabney, I. 2001. *Netlab Algorithm for Pattern Recognition*. Advances in Pattern Recognition. Springer.

Neal, R. and G. Hinton 1999. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. Jordan (Ed.), *Learning in Graphical Models*. Kluwer.

Oliver, J. and R. Baxter 1995, August. MML and Bayesianism: similarities and differences. Technical Report 206, Department of Computer Science, Monash University, Australia.

Oliver, J., R. Baxter, and C. Wallace 1996. Unsupervised learning using MML. In *Proc. of the Thirteenth International Conference on Machine Learning*, pp. 364–372. San Francisco: CA: Morgan Kaufmann.

Redner, R. and H. Walker 1984. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review* **26** (2).

Rissanen, J. 1986. Stochastic complexity. *Annals of Statistics* **14**, 1080–1100.

Ritter, H., T. Martinetz, and K. Schulten 1992. *Neural Computation and Self-Organizing Maps: An Introduction.* Addison-Wesley, Reading, MA.

Roberts, G., G.J.Myatt, W. Johnson, K. Cross, and P. Blower 2000. Leadscope. *Journal of Chemical Information and Computer Science* **40** (6), 1302.

Sahami 1998. *Using Machine Learning to Improve Information Access.* Ph.D. thesis, Stanford University.

Sammon, J. 1969. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers* **C-18** (5), 401–409.

Schwarz, G. 1978. Estimating the dimension of a model. *Annals of Statistics* **6**, 461–464.

Seber, G. and C. Wild 1989. *Nonlinear Regression.* New York, NY: John Wiley and Sons.

Spath, H. 1980. *Cluster Analysis Algorithm.* Chichester, West Sussex: Ellis Horwood.

Streit, R. and T. Luginbuhl 1994. Maximum likelihood training of probabilistic neural networks. *IEEE Trans. on Neural Networks* **5** (5), 764–783.

Svensén, M. 1998. *GTM: The Generative Topographic Mapping.* Ph.D. thesis, Aston University, Birmingham, UK.

Tibshirani, R. 1992. Principal curves revisited. *Statistics and Computation* **2**, 183–190.

Tiňo, P. and I. Nabney 2002. Hierarchical GTM: constructing localized nonlinear projection manifolds in a principled way. *IEEE Transaction on Pattern Analysis and Machine Intelligence* **24**.

Tipping, M. and C. Bishop 1999. Mixtures of probabilistic principal component analysers. *J. Roy. Statist. Soc.* **B 61**, 611–622.

Tipping, M. and D. Lowe 1997, July. Shadow targets: a novel algorithm for topographic projections by radial basis function. In *Artificial Neural Networks*, Number 440. IEE.

Titterington, D., A. Smith, and U. Makov 1985. *Statistical Analysis of Finite Mixture Distributions.* John Wiley & Sons.

van Rijsbergen, C. 1979. *Information Retrieval.* London: Butterworth.

Vlassis, N. and A. Likas 1999. A kurtosis-based dynamic approach to gaussian mixture modeling. *IEEE transactions on Systems Man and Cybernetics Part A- Systems and Humans* **29**, 393–399.

Wallace, C. and D. Dowe 1999. Minimum message length and Kolmogorov complexity. *The computer Journal* **42**, 270–283.

Wallace, C. and P. Freeman 1987. Estimation and inference by compact coding. *Journal of the Royal Statistical Society B* **49** (3), 241–252.

Zilliox, H. 1998. Quality Control of High Throughput Screening. Master's thesis, Aston University.