DATA MINING AND VISUALIZATION

César García-Osorio

A thesis submitted in partial fulfilment of the requirements of the University of Paisley for the degree of Doctor of Philosophy

January 2005

Abstract

This thesis represents the outcome of PhD studies in the task of using visual means for identifying structure in high dimensional data sets. Our basic method is the development of algorithms which will run on computers which will display the data in such a way that a human is most easily able to identify some form of structure in the data set.

The first group of algorithms is based on artificial neural networks and has to do with the automatic identification of the direction which best shows the clusters in the data. This kind of artificial neural networks are usually called Exploratory Projection Pursuit neural networks (EPP neural networks). The second group of algorithms uses a transformation of the data set. Instead of projection of high-dimensional points, this time the coordinates of these points are used to obtain a function. This way every point gives a curve. These curves are called Andrews' curves.

The new work in the thesis starts, in Chapter 3, with a comparison of the Maximum Likelihood and the Output Function EPP neural networks. We show that, while the former is faster, the latter is more accurate. We then show how to combine these algorithms to obtain a new one that is both fast and accurate. We measure the performance of the three algorithms using an artificial data set with known structure but also using a real data set. The comparison using real data sets is based on a novel use of cluster validation indices and gives a framework for comparison that can be used for other EPP algorithms. To our knowledge that is something that has not been done before. We also improve the convergence and stability of EPP neural networks by using Andrews' curves as initializations mechanism.

In Chapter 4 we give new perspective of Wegman and Shen's curves, an extension of Andrews's curves for obtaining a bi-dimensional grand tour. We give our own extension suitable for obtaining a three-dimensional grand tour. Instead of a curve, for each point, our extension gives three surfaces. We investigate the properties of our surfaces and exemplify their use as an exploratory data analysis tool (see `http://pisuerga.inf.ubu.es/cgosorio/Andrews/exploratoryPage.html`). We also show, in Chapter 5, how the Self-organizing Map of Kohonen can be used to help our surfaces identify structure in high-volume data sets and also, on the other hand, how our surfaces can be used to visualize the structure captured by Self-organising Maps.

In the last part of the thesis we investigate the use of the surfaces as a visualization mechanism of the feature space which is found in kernel methods. We can take advantage of the non-linearities introduced by the kernel matrix, to identify difficult clusters, which without them we could not identify. Also, we propose the use of bootstrapping and bagging to regain the sparseness found in Support Vector Machines in the context of Kernel Principal Component Analysis.

Acknowledgments

First and foremost I want to thank to my supervisor Colin Fyfe, his expert guidance, his patience and his stimulating ideas have been the basement onto which this thesis has been built.

Another important ingredient in the realization of this thesis has been the good atmosphere created by my colleagues in the University of Paisley: Jos Koetsier, Donald McDonald, Ying (Hannah) Han, Lina Petrakieva, Stephen McGlinchey, Gayle Leen, Douglas Wang, Marian Peña, Pablo Marin-Franch. Thanks to all of them. They have make me feel as at home. Thanks also to Nicolas García-Pedraja for the last minutes assistance.

There have been some other people who, at some way or other, have helped me in this journey but are left unmentioned. I hope I can name them on other occasions.

Finally, thanks to my parents, my sister, my brother, my family and friends for their positive attitude, understanding and support. This thesis is dedicated to them.

Contents

Chapter 1

Introduction

1.1 Visualization

The advent of the personal computer has provided mankind with enormous benefits — we have access to more information than ever before and have such access virtually 24 hours per day, 7 days per week thanks to the internet. However it is in the nature of humankind that we always wish for more: in this case, we have all this data but actually finding information within the data is often an extremely complex task. This thesis will devote itself to this problem. We will restrict ourselves to numeric data: this is the simplest and perhaps most frequent form of data in which we seek information. The numeric data will often be of high-dimensionality: we might envisage information about an individual consisting of his height, weight, bank balance, etc. so that each of these fields constitutes one dimension of a long description of the individual. Thus, if we have 10 fields, we have a 10-dimensional vector describing each individual. The task then, might be to identify groups of individuals all of whom share some common characteristic. For example, it is known that tall people tend to earn more than their shorter brethren (a fact which causes some of us some disquiet). In order to ascertain this fact, we must have a way of identifying structure across dimension boundaries. This thesis will investigate methods for performing such identifications in a semi-automated manner.

We state semi-automated since it will be one of the themes of the thesis that the computer and the human both have roles to play in identifying structure: the computer is very good at handling large volumes of data and manipulating such data in an automatic manner; but humans are very good at pattern identification

— much better indeed than computers (consider how face recognition systems have failed to live up to our expectations, even now). Therefore, we envisage a partnership between the human and the computer software with each performing the role to which he/it is best suited. The computer software will manipulate the high-dimensional data and present it to the human in a way which facilitates the human's pattern matching. An example of this is Exploratory Projection Pursuit (see Chapter 3) in which the high-dimensional data is projected onto a two dimensional subspace in such a way that the structure of the data (for example, clusters) is most easily identified by a human by eye. Another example is the use of Andrews' Curves in which each data point is represented by a curve. A human can run his eye along a set of curves (representing the members of the data set) and identify particular regions of the curve which are optimal for identifying clusters in the data set. Of interest in this context, is our extension in which a moving three-dimensional image is created in which we can see clouds of data points moving as we move along the curves; in a very real sense, the data which dance together are members of the same cluster.

However, interestingly, several of the methods which we use to find structure (i.e. the computer software part of the partnership) are based on neural networks, the network of neurons which we have in our brains. Does this suggest that humans might be able to dispense with the computer software and perform the whole task themselves? It is a nice thought but common experience suggests that it is not so; we require intelligent software to help us find structure in high-dimensional data sets. Therefore we will first discuss this aspect of our work before discussing the whole work in general.

1.2 Artificial Neural Networks

Our model of the single neuron is fairly simple processing device: it is intended to be an abstraction of the neuron which exists in vivo (see Figure 1.1). We consider the inputs to the neuron to be floating point numbers; this corresponds to the electro-chemical inputs which a neuron will receive from other neurons or from transducers which take in signals from the environment. The effect of such signals is modulated by the efficiency of the transmission mechanisms which in real neurons is determined by the characteristics of the synapses. We will model an efficient synapse with a large magnitude weight while a less efficient synapse will have a smaller magnitude weight. Since some neurons are inhibitory and some excitatory, we will have positive and negative weights. At the body of the

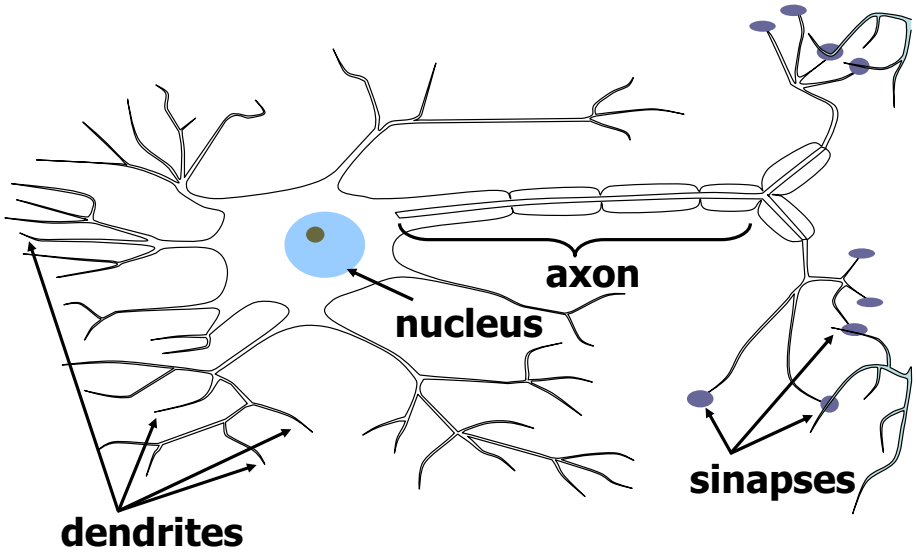


Figure 1.1: Representation of a typical neuron.

neural cell, a summation of the inputs is performed. We will typically model this by

$$y = \sum_j w_j x_j \tag{1.1}$$

so that we are summing the weighted inputs, with the weights w_j corresponding to the synaptic efficiencies as described above. Often there may be a nonlinearity involved so that

$$y = f\left(\sum_j w_j x_j\right) \tag{1.2}$$

but still, the individual artificial neuron is a fairly simple building block.

The major aspect of artificial neural networks which makes them useful machines is their ability to re-parameterize in relation to specific data sets: the actual parameters (for example, the w_j above) are changed in response to the statistics of the input data. This is most often performed in an incremental manner. For example, the Hebbian learning rule can be formulated as

$$\Delta w_j = \eta y x_j \tag{1.3}$$

$$w_j = w_j + \Delta w_j \tag{1.4}$$

The first equation calculates the change in the weights which is based on the current input, x_j, and the neuron's response to that, y. The η term is known as

the learning rate and is typically a small value (perhaps, 0.001) which is often annealed to zero during the course of the simulation. The second equation calculates the new value of the weight, w_j using this change in weights. Together these two equations define on-line learning.

We also at this stage note that we used no training signal in this rule: we did not give the rule any indication of what we expected from the trained network. This will be typical of the networks which we use in this thesis: we are mainly interested in unsupervised learning. Many artificial neural networks use supervised learning or learning with a teacher. Since we are working in the field of exploratory data analysis, we wish the networks we use to self-organize in order to extract specific types of structure from the data sets. Therefore, unsupervised artificial neural networks will predominate in this thesis and will be met throughout the thesis. We next discuss the overall structure of the thesis.

1.3 Structure of the Thesis

We begin the thesis with a review in Chapter 2 of related work performed by others. We begin with a short description of two taxonomies of visualization methods and note that there is no one universally accepted taxonomy which does justice to the breadth of work available to researchers today. Therefore we establish our own taxonomy on which the remainder of this chapter is based. We then discuss various linear projection methods and a variety of topology preserving methods, some of which are based on artificial neural networks. We then consider several methods from the literature which are important in representing data. Perhaps of most importance in this section is Andrews' Curves [2] which we later extend in Chapter 4. We also briefly discuss the Grand Tour [4] — looking at a high-dimensional data set in all possible low-dimensional projections.

In Chapter 3, we first perform a comparison between two existing neural network methods of performing Exploratory Projection Pursuit (EPP) [47], the Maximum Likelihood [28] and the Output Function [56] EPP neural networks. We show that the methods can be combined and that the resulting algorithm is very fast, accurate and stable. We also show how the most recent of the two methods, the Maximum Likelihood EPP neural network, can be extended to search for a type of structure, skewness, which it had previously been unable to find. Finally, we use Andrews' Curves to initialize the Exploratory Projection Pursuit network's weights which facilitates the convergence of the algorithm for

Output Function EPP neural network but emphasizes the innate inaccuracy of the Maximum Likelihood EPP neural network.

In Chapter 4, we extend the work of Andrews [2] and of Wegman and Shen [179] and, in so doing, create a representation of the data which enables a three-dimensional motion which humans find assists in cluster identification. We illustrate the method with brushing on a real data set and discover clusters which the human researchers who provided us with the data set had not previously identified. We also, in Chapter 5, show how Andrews' curves can be used to identify the structure found by self-organizing maps and, alternatively, how to enhance Andrews' Curves using the self-organizing map, and how to combine in the same display two different perspectives of the same data. We end this chapter proposing a generalization of our extension to Andrews' curves.

Chapter 6 considers the problem of identifying structure in high-dimensional feature spaces: kernel methods are a group of methods which have become popular recently for regression, classification, clustering, etc.... They are all based on the kernel trick which may be characterized by stating that provided we can calculate the scalar product of two points in the feature space, we need never know the coordinates of the point itself. This is a somewhat amazing fact but leaves us with little intuition about the nature of the space itself. Thus we investigate such feature spaces using Andrews' Curves and our extensions thereof. We also develop a bootstrapping method which allows us to find sparse representations of a data set with Sparse Kernel Principal Component Analysis [155], something which existed in Vapnik's original Support Vector Machines [168] but which was absent in the original Kernel Principal Component Analysis [148].

1.4 Contribution of the Research

The specific contributions of the research presented in this thesis are:

- We perform a comparative study of two existing Exploratory Projection Pursuit neural networks, the Maximum Likelihood and the Output Function EPP neural networks, and show that the first converges faster while the second is more accurate.

- We show how to combine the Maximum Likelihood and the Output Function algorithms and that this combination gives us the best of both worlds. That is, a new algorithm that is both fast and accurate.

- We extend the Maximum Likelihood EPP algorithms so that it can now search for structure (skewness in a data set) which it was previously unable to detect.
- In the context of cluster validation there exists a set of indices which have been developed to compare clustering methods. We have used, in a novel manner, these indices to construct a comparison framework for EPP algorithms.
- We show how to initialize the EPP neural networks using Andrews' curves and that this initialization is extremely beneficial to convergence.
- We show that the Maximum Likelihood is inherently inaccurate in that, even when the parameters are set to their optimal values, they will still diverge from these within the algorithm.
- We extend Andrews' Curves and show how the new three dimensional representation facilitates cluster identification.
- We investigate the properties of the extension, such as, distance and mean preservation.
- We compare the extension with that performed by Wegman and colleagues, particularly with respect to their abilities to search the whole space.
- Investigating the structures found by Self-organizing Maps is an on-going research problem. We are the first to use Andrews' Curves to perform this task.
- An existing problem with Andrews' Curves and its extensions is the fact that these methods are only applicable to low volumes of data. We show how using the centres of a trained Self-organizing Map circumvents this problem.
- We use Andrews' Curves and our extensions to investigate the structure of high- (perhaps infinite-)dimensional feature spaces.
- The literature contains two extremes with respect to Kernel Principal Component Analysis: at one end, we have standard Kernel Principal Component Analysis which uses all the data points at the expense of losing sparsity which we find in Support Vector Machines; at the other end, we have

Sparse Kernel Principal Component Analysis which uses a single data point to represent the "Kernel Principal Component" direction. We have created a method using bootstrapping which recovers the sparsity of Support Vector Machines in the context of Kernel Principal Component Analysis.

Many of these contributions have already been published in [61-69]. We begin with a review of the existing literature on visualization.

Chapter 2

Literature Review

The roots of information visualization as a practical field can be established in the works of Tukey [167], Bertin [11, 12] and Tufte [166] who focused on 2D and 3D visualization and produced general rules for the plane, the colour composition and attribute mapping, among others. The use of the attributes of a database as dimensions was the rationale behind the study of the multidimensional techniques. The work presented in this thesis can be classified into this last category. However, the study of multivariate visualization began some centuries before; following Wong and Bergeron [183], the evolution of the field can divided into four periods:

1. *The Searching Stage* (from 1782 to 1976): characterized by relatively small sized data, and tools for data visualization that usually consisted of colour pencils and graph paper.

2. *The Awakening Stage* (from 1977 to 1985): two and three dimensional spatial data were the most common data types being studied, although multivariate data started gaining more attention.

3. *The Discovery Stage* (from 1986 to 1991): the limited availability of high speed graphics hardware during the previous stage was gradually conquered. Most of the visualization methods presented in this chapter were developed in this period.

4. *The Elaboration and Assessment Stage* (from 1992 to present): this period has seen a retrenchment in the development of new visualization techniques. Some of the most recently developed tools are elaborations of work done in previous stages.

To put in context the visualization methods introduced in this thesis, in this chapter, we comment on some of the existing visualization methods. A full review of the entire field is beyond the scope of this chapter. A general overview can be found in the following references: Wong and Bergeron [183], Ferreira de Oliveira [45] and Keim [98, 96]. We have preferred to concentrate on methods more directly related to the ones proposed in the thesis, and in turn on some of the methods related with those. The depth in which the methods are presented is also motivated by how much we think the methods are related to our visualization methods.

We start the chapter with a short description of some taxonomies of visualization methods, then we present our own vision of the field that we have used to structure the remainder of the chapter. After that we explain in detail each class in our classification emphasizing some of the more significant visualization methods belonging to that class.

We will finish by giving a list of some of the software tools for data visualization freely available on the Internet.

2.1 Taxonomies of Visualization Methods

There is no universal consensus on the best taxonomy. In the following subsections we present two of the more elaborate ones. Then we explain the classification used in this chapter.

2.1.1 Taxonomy of Techniques by Shneiderman

Shneiderman, in [151], propounded a task by data type taxonomy of information visualizations. The seven tasks are:

Overview: Gain an overview of the entire collection.

Zoom: Zoom in on items of interest allowing a more detailed view.

Filter: Filter out uninteresting items reducing the size of search.

Details-on-demand: Select an item or group and get details when needed.

Relate: View relationships among items

History: Keep a history of actions to support undo, replay, and progressive refinement allowing a mistake to be undone, or a series of steps to be replayed.

Extract: Allow extraction of sub-collections and saving, printing or dragging to another application.

All these tasks can be implemented with the software developed for this thesis.

The seven data types considered are:

1-dimensional: linear data types include textual documents, program source code, and lists of names in alphabetical order.

2-dimensional: planar or map data include geographic maps, floor plans, or newspaper layouts.

3-dimensional: real-world objects such as molecules, the human body, and buildings. The user must cope with understanding their position and orientation when viewing the objects.

Multi-dimensional: items with n attributes become points in a n-dimensional space.

Temporal: this data type is different from the 1-dimensional data type because now the items have a start and finish time and items may overlap.

Tree: collections of items with each item having a link to one parent item (except the root). Items and the links between parent and child can have multiple attributes.

Network: the items are linked to an arbitrary number of other items and the relationships among items cannot be captured with a tree structure.

Although the main data type in which are interested in this thesis is the multi-dimensional, Andrews' curves have been used by Embrechts et al. [44] to represent time series, and it seems likely that the extension of Andrews' curves proposed in Chapter 4 could be used on that kind of data as well.

2.1.2 Taxonomies of Techniques by Keim

Keim and Kriesel, in [100], categorize the visualization techniques into the following:

- **Geometric Techniques**. The aim of those techniques is to find 'interesting' transformations of multidimensional data sets. This class includes Scatterplot Matrices [9, 26], Principal Component Analysis [91], Factor Analysis [116], Multidimensional Scaling [165, 149, 106], Projection Pursuit [81], Parallel Coordinates [89, 174], etc.

- **Icon-Based Techniques**. Every multidimensional item is mapped onto an icon or glyph. The amount of observations that it is possible to visualize at the same time is quite limited, and depends on the characteristics of the icon. Examples are: Chernoff Faces [23], Star Icons [172], Colour Icons [115], Stick Figure Icons [75]. Later, we will give more details about the first of these methods.

- **Pixel-oriented Techniques**. In these techniques, the data attributes are mapped onto pixels; the colour of every pixel depends on the values of the attributes. An important aspect is the spatial distribution of the pixels; some examples are the recursive pattern technique, the circle segments technique and the spiral technique (all these methods are explained in [99]).

- **Hierarchical Techniques**. The multidimensional space is divided hierarchically in subspaces. Examples are Dimensional Stacking [108] and Worlds-within-Worlds [10].

- **Graph-based Techniques**. These are specialized techniques for presenting large graphs using specific layout algorithms, query languages and abstraction techniques. For example, the Treemap [150].

The previous classification is extended by Keim [97] with another two orthogonal criteria: the distortion technique and the interaction technique. The first is to show portions of the data with different levels of detail. The second allows the user to directly interact with the visualization. The three criteria are shown as orthogonal axes of a classification system as we can see in Figure 2.1. It is worth noting that the distortion techniques alone deserve their own taxonomy that has been analyzed by Leung and Apperley [114].

In [98], the interaction and distortion criteria are combined in one axis, the data to be visualized is added as a new criteria and the visualization techniques are presented slightly changed as we can see in Figure 2.2.

The methods proposed in this thesis fall into the geometric techniques category of [97], or similarly in the geometrically transformed display category used in [98].

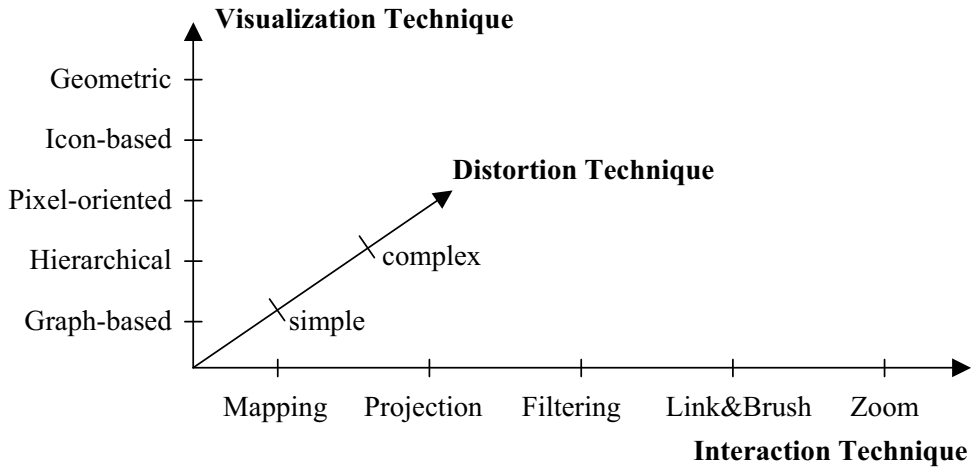


Figure 2.1: The classification of information visualization techniques as it appears in [97].

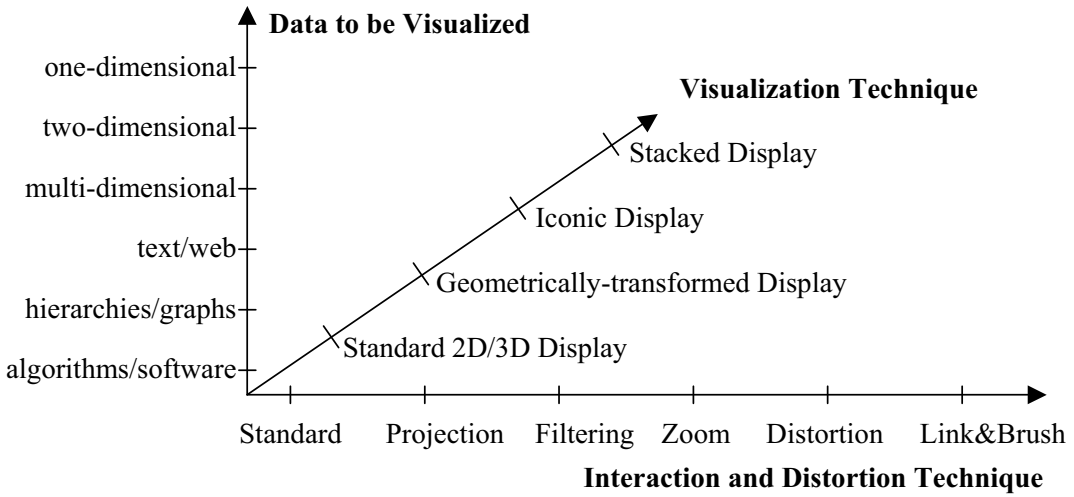


Figure 2.2: The classification of information visualization techniques as it appears in [98].

2.1.3 Our own taxonomy

We have grouped the visualization methods presented in this chapter according to the following classification:

- **Linear Projection Methods**. A lower dimensional representation of the data is obtained using a linear projection of the high-dimensional space. Some examples are Principal Component Analysis (PCA) [90], Exploratory Projection Pursuit (EPP) [47] and Scatterplot Matrix [21].

- **Topology Preservation Methods**. A non-linear mapping between the high-dimensional space and the lower-dimensional representation is used trying to maintain the topology of the high-dimensional space: Sammon Mapping [140], Curvilinear Component Analysis (CCA) [33], Self Organizing Maps (SOM) [104] and Spring Methods [20, 125].

- **High-Dimensional Representations**. This time there is no dimension reduction; instead of that, all the values of the coordinates are used to obtain the graphical representation; examples are Chernoff Faces [23], Parallel Coordinates [89, 174], Andrews' Curves [2] and Multidimensional Stacking [108].

- **Grand Tour Methods**. Instead of one static representation of the data, a sequence of projections is used to study the structure of the data. Some of the algorithms used to obtain such a sequence are the Asimov-Buja Winding Algorithm, the Random Curve Algorithm, the Fractal Curve Algorithm and the pseudo Grand Tour (details of all these can be found in [4, 179, 180]).

2.2 Linear Projection Methods

Sometimes it is possible to see the structure of a high-dimensional data set just by changing the basis of the considered space. However, the decision as to what basis to use would require a fore-knowledge of the pattern one wants to identify. One possible solution to this problem is the use of the directions which explain most of the variance in the data set; such directions are called the *principal component directions* of the data set. The rationale for this is that without any information about the distribution of the data set, the standard assumption is to suppose that the data set follows a Gaussian distribution (see Figure 2.3),

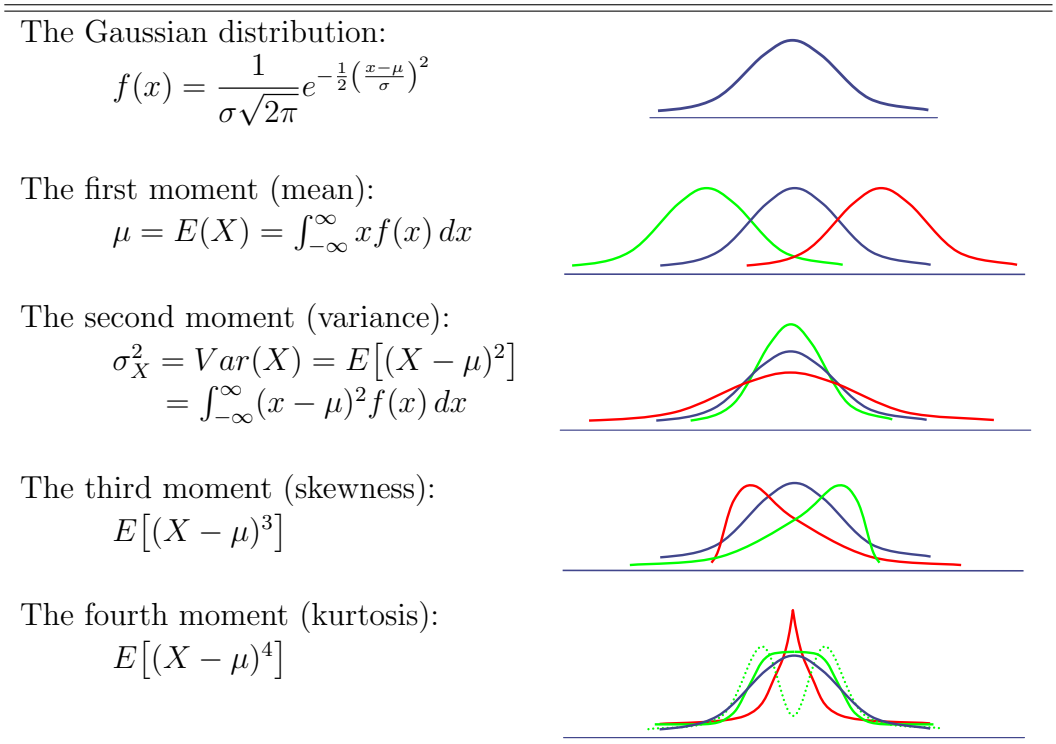


Figure 2.3: The Gaussian distribution and its higher order moments.

and in Gaussian distributions the amount of information is directly proportional to the variance. Even if the distribution is not Gaussian there is liable to be more information in high variance projections than in a low variance projection. The use of the principal component projections to analyze a data set is known as Principal Component Analysis (or PCA) [80]. When the distribution is not a Gaussian, instead of maximizing the variance, we can try to maximize other moments of the distribution[1] (Figure 2.3). This last approach gives rise to a set of techniques known as Exploratory Projection Pursuit [48, 47, 92]. In the following sections we give more details about these two techniques.

2.2.1 Principal Component Analysis

For some data sets the projection onto the principal components lets us visualize clearly their structure. In this way we can try to discover the structure of a

[1]Depending on the values of these moments we can speak about *skewed* distributions when distributions are not symmetric; *leptokurtotic* or *super-Gaussian* distributions when distributions are more kurtotic than a Gaussian distribution, and conversely *platykurtotic* or *sub-Gaussian* distributions when distributions are less kurtotic than a Gaussian distribution.

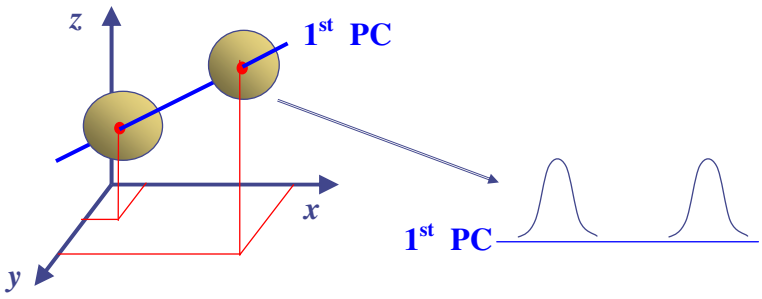


Figure 2.4: An example in which the projection onto the PC directions reveals for us structure in the data set (in this case, the existence of two clusters).

high dimensional data set by projecting it onto the subspace defined by the first principal components (usually the first two or three). The projection onto the first principal component contains more variance than any other projection, then the projection onto the second principal component, which is orthogonal to the first one, contains the next most, and so on. In Figure 2.4 we can see an example of this. We have a data set with three variables (in this case the dimensionality of the data set makes a graphical representation possible but with greater than three dimensions this is not the case), and we can discover the existence of the clusters when we project it onto the one dimensional space of the first principal component (actually in this case we can see the histogram of the projection).

The principal components represent the most important linear characteristics of the data in terms of minimizing the mean square error. The aim of PCA is the identification of the dependency existing between the variables (see Jolliffe [90] for an extensive treatment of this technique). The higher the correlation between the variables is, the fewer the number of the independent variables is and the more the dimensionality can be reduced.

Using PCA a high dimensional data set can be represented by a few principal components (from the point of view of graphical representation, the ideal number is two or three), so PCA can be considered as much a feature extraction technique as a data compression technique.

How to calculate the PCs

There exist several algebraic methods to calculate the PCs of a data set: the power method, the Householder transformation, the Lanczos method, ... and also neural networks algorithms; a good review of which is presented by Dia-

mantaras and Kung [37]. The most famous of the neural network methods is Oja's algorithm [129, 130, 131, 132]; in this thesis we are interested in a negative feedback implementation of PCA defined by (2.1)-(2.3) [52, 51]. Let us have an N-dimensional input vector, $\mathbf{x}$, and an M-dimensional output vector, $\mathbf{y}$, with w_{ij} being the weight linking the j^{th} input to the i^{th} output. The learning rate, η is a small value which will be annealed to zero over the course of training the network. The activation passing from input to output through the weights is described by (2.1). The activation is then fed back through the weights from the outputs and the error e calculated for each input dimension. Finally the weights are updated using simple Hebbian learning (see Section 1.2).

$$y_i = \sum_{j=1}^{N} w_{ij} x_j, \forall i \tag{2.1}$$

$$e_j = x_j - \sum_{i=1}^{M} w_{ij} y_i \tag{2.2}$$

$$\Delta w_{ij} = \eta e_j y_i \tag{2.3}$$

Several variations of this network have been used to perform clustering with topology preservation [53], to perform Factor Analysis [58, 22] and to perform Exploratory Projection Pursuit [55, 54].

2.2.2 Exploratory Projection Pursuit

Sometimes the projection onto the principal components does not give any clue about the structure of the data set; in Figure 2.5 we can see that this time the projection onto the first PC direction (that is approximately the x axis direction) does not give any information about the structure in the data set. Something similar happens with the projection onto the y axis. On the contrary, the projection onto the z axis reveals the existence of two clusters to us. In circumstances like this we can use Exploratory Projection Pursuit.

Exploratory Projection Pursuit (see, for example, Friedman [47]) is a generic name for the set of techniques designed to identify structure in high dimension data sets. In such data sets, structure often exists across data field boundaries and one way to reveal such structure is to project the data onto a lower dimension space and then look for structure in this lower dimension projection by eye. However we need to determine what constitutes the best subspace onto which the

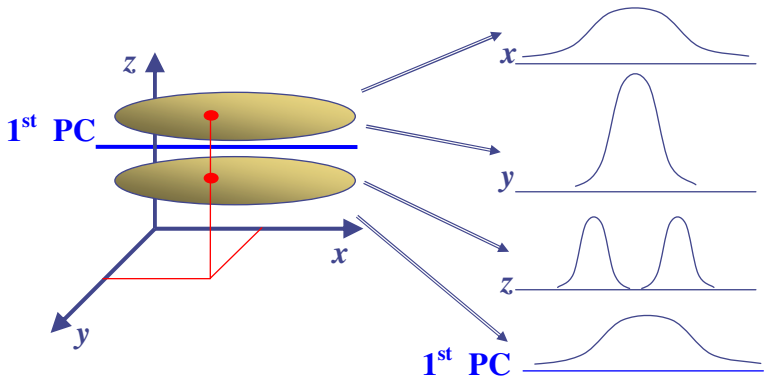


Figure 2.5: An example in which the projection onto the first PC does not give information about the structure of the data set.

data should be projected.

Exploratory Projection Pursuit (EPP) is a technique for exploring high dimensional spaces. As we have seen, PCA searches for filters in a data set which maximise the variance of the projections of the data onto these filters. EPP can be thought of as an extension of PCA since it is a technique for finding projections which maximise some statistic over the data set. For example, when we wish to identify clusters in a data set, their presence might be revealed by a negative fourth moment (kurtosis) in the projections. EPP has also been implemented using artificial neural networks (see Fyfe and Baddeley [55] and Fyfe [54]).

As noted by Diaconis and Freedman [36], the typical projection of high dimensional data will have a Gaussian distribution and so little structure will be evident. This has led researchers to suggest that what they should be looking for is a projection which gives a distribution as different from a Gaussian as possible. Thus we typically define an index of "interestingness" in terms of how far the resultant projection is from a Gaussian distribution. Since the Gaussian distribution is totally determined by its first two moments, we usually sphere the data (make it zero mean and with covariance matrix the identity matrix) so that we have a level playing field to determine departures from Gaussianity. In this section, we will review two methods of performing Exploratory Projection Pursuit with neural algorithms based on the network introduced in the previous section. In Chapter 3, we propose a new method which is a combination of these two and compare the three methods.

The Output Functions EPP Algorithm

Two common measures of deviation from a Gaussian distribution are based on the higher order moments of the distribution (see Figure 2.3). Skewness is based on the normalized third moment of the distribution and basically measures if the distribution is symmetrical. Kurtosis is based on the normalized fourth moment of the distribution and measures the heaviness of the tails of a distribution. A bimodal distribution will often also have a negative kurtosis and therefore kurtosis can signal that a particular distribution shows evidence of clustering. Whilst these measures have their drawbacks as measures of deviation from normality (particularly their sensitivity to outliers), their simplicity makes them ideal for explanatory purposes.

The only difference between the PCA network and the EPP network is that a function of the output activations is calculated and used in the simple Hebbian learning procedure. We have for N dimensional input data and M output neurons

$$y_i = \sum_{j=1}^{N} w_{ij} x_j \tag{2.4}$$

$$e_j = x_j - \sum_{i=1}^{M} w_{ij} y_i \tag{2.5}$$

$$r_i = f\left(\sum_{j=1}^{N} w_{ij} x_j\right) = f(y_i) \tag{2.6}$$

$$\Delta w_{ij} = \eta_t r_i e_j \tag{2.7}$$

$$= \eta_t f\left(\sum_{k=1}^{N} w_{ik} x_k\right)\left\{x_j - \sum_{l=1}^{M} w_{lj} \sum_{p=1}^{N} w_{lp} x_p\right\} \tag{2.8}$$

where r_i is the value of the function $f(.)$ on the i^{th} output neuron. Thus (2.8) may be written in matrix form as

$$\Delta \mathbf{W}(t) = \eta(t)[\mathbf{I} - \mathbf{W}(t)\mathbf{W}^T(t)]\mathbf{x}(t) f(\mathbf{x}^T(t)\mathbf{W}(t)) \tag{2.9}$$

where t is an index of time, $\mathbf{I}$ is the $(M \times M)$ identity matrix, $\mathbf{W}(t)$ is a $(M \times N)$ matrix and $\mathbf{x}(t)$ is a $(M \times 1)$ column vector.

Following Karkunen and Joutsensalo [93], we can derive (2.9) as an approximation to the maximization of a function of the weights $J(\mathbf{W}) = \sum_{i=1}^{M} E(g[\mathbf{x}^T \mathbf{w}_i] | \mathbf{w}_i)$ with $E(.)$ the expectation operator and $\mathbf{w}_i$ the weight vector into the i^{th} output

neuron.

We must ensure that the optimal solution is kept bounded; otherwise there is nothing to stop the weights from growing without bound. Formally,

$$\text{let } J(\mathbf{W}) = \sum_{i=1}^{M} E(g[\mathbf{x}^T\mathbf{w}_i]|\mathbf{w}_i) + \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\lambda_{ij}[\mathbf{w}_i^T\mathbf{w}_j - a_{ij}] \tag{2.10}$$

where the last term enforces the constraints $\mathbf{w}_i^T\mathbf{w}_j - a_{ij}$ using the Lagrange multipliers λ_{ij}. As usual, we differentiate this equation with respect to the weights and with respect to the Lagrange multipliers. This yields respectively, at a stationary point,

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}} = E(\mathbf{x}g'(\mathbf{x}^T\mathbf{W})|\mathbf{W}) + \mathbf{W}\mathbf{\Lambda} = 0 \text{ and} \tag{2.11}$$

$$\mathbf{W}^T\mathbf{W} = \mathbf{A} \tag{2.12}$$

where $g'(\mathbf{x}^T\mathbf{W})$ is the elementwise derivative of $g(\mathbf{x}^T\mathbf{W})$ with respect to $\mathbf{x}^T\mathbf{W}$, $\mathbf{A}$ is the matrix of parameters a_{ij} (often the identity matrix) and $\mathbf{\Lambda}$ is the matrix of Lagrange multipliers. Equations (2.11) and (2.12) define the optimal points of the process. Pre-multiplying (2.11) by $\mathbf{W}^T$ and inserting (2.12), we get

$$\mathbf{\Lambda} = -\mathbf{A}^{-\mathbf{1}}\mathbf{W}^T E(\mathbf{x}g'(\mathbf{x}^T\mathbf{W})|\mathbf{W})$$

and using this value and reinserting this optimal value of $\mathbf{\Lambda}$ into (2.11) yields the equation,

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}} = [\mathbf{I} - \mathbf{W}\mathbf{A}^{-1}\mathbf{W}^T]E(\mathbf{x}g'(\mathbf{x}^T\mathbf{W})|\mathbf{W}) \tag{2.13}$$

We wish to use an instantaneous version of this in a gradient ascent algorithm

$$\Delta\mathbf{W} \propto \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$$

to yield

$$\Delta\mathbf{W} = \mu[\mathbf{I} - \mathbf{W}\mathbf{A}^{-1}\mathbf{W}^T]\mathbf{x}g'(\mathbf{x}^T\mathbf{W}) \tag{2.14}$$

We will be interested in the special case where the $\mathbf{W}$ values form an orthonormal basis of the data space and so $\mathbf{A} = \mathbf{I}$, the identity matrix. Therefore, we can equate (2.14) with (2.9).

Summing up, the network operation is

Feedforward:

$$y_i = \sum_{j=1}^{N} w_{ij} x_j, \forall i$$

Feedback:

$$e_j = x_j - \sum_{i=1}^{M} w_{ij} y_i$$

Weight change:

$$\Delta w_{ij} = \eta f(y_i) e_j$$

where the function $f(.)$ is determined by the function $g(.)$ which we wish to maximise.

The Maximum Likelihood EPP Algorithm

Various researchers, e.g. Xu [184] and Karhunen and Joutsensalo [93], have shown that the learning rules (2.1)-(2.3) can be derived as an approximation to the best linear compression of the data. Thus we may start with the cost function

$$J = 1^T E\{(\mathbf{x} - \mathbf{W}\mathbf{y})^2\} \tag{2.15}$$

which we minimize to get (2.3).

We may show that the minimization of J is equivalent to minimizing the negative log probabilities of the residual, $\mathbf{e}$, if $\mathbf{e}$ is Gaussian and thus is equal to maximizing the probabilities of the residual (see Bishop [13]). Let the probability of the residuals $p(\mathbf{e}) = \frac{1}{Z}\exp(-\mathbf{e}^2)$ where Z is a normalization term. Then we can denote a general cost function associated with the network as

$$J = -\log p(\mathbf{e}) = (\mathbf{e})^2 + K \tag{2.16}$$

Intuitively, J gives a measure as to how probable the residual is under the current model parameters. Since the only parameters which can be changed are the weights which both feed forward and backward, these are the parameters which we must adapt in order to make the residuals more likely under the model. Therefore performing gradient descent on J we have

$$\Delta \mathbf{W} \propto -\frac{\partial J}{\partial \mathbf{W}} = -\frac{\partial J}{\partial \mathbf{e}} \frac{\partial \mathbf{e}}{\partial \mathbf{W}} \approx 2\mathbf{y}\mathbf{e}^T \tag{2.17}$$

where we have discarded a relatively unimportant term [93].

An extension of the above was considered by Corchado and Fyfe [59] and Fyfe and McDonald [27] with a more general cost function

$$J = f_1(e) = f_1(\mathbf{x} - \mathbf{W}\mathbf{y}) \tag{2.18}$$

Let us now consider the residual after the feedback to have probability density function

$$p(\mathbf{e}) = \frac{1}{Z}\exp(-|\mathbf{e}|^p) \tag{2.19}$$

Then we can denote a general cost function associated with this network as

$$J = -\log p(\mathbf{e}) = |\mathbf{e}|^p + K \tag{2.20}$$

where K is a constant. Therefore performing gradient descent on J we have

$$\Delta\mathbf{W} \propto -\frac{\partial J}{\partial \mathbf{W}} = -\frac{\partial J}{\partial \mathbf{e}}\frac{\partial \mathbf{e}}{\partial \mathbf{W}} \approx p\mathbf{y}\big(|\mathbf{e}|^{p-1}\operatorname{sign}(\mathbf{e})\big)^T \tag{2.21}$$

We would expect that for leptokurtotic residuals (more kurtotic than a Gaussian distribution), values of $p < 2$ would be appropriate, while platykurtotic residuals (less kurtotic than a Gaussian), values of $p > 2$ would be appropriate. It has been shown (Hyvarinen et al. [84]) that it is less important to get exactly the correct distribution when searching for a specific source than it is to get an approximately correct distribution i.e. all supergaussian signals can be retrieved using a generic leptokurtotic distribution and all subgaussian signals can be retrieved using a generic platykurtotic distribution.

Therefore the network operation is:

Feedforward:

$$y_i = \sum_{j=1}^{N} w_{ij}x_j, \forall i$$

Feedback:

$$e_j = x_j - \sum_{i=1}^{M} w_{ij}y_i$$

Weight change:

$$\Delta w_{ij} = \eta y_i \operatorname{sign}(e_j)|e_j|^{p-1}$$

where the value of p is determined by the type of structure we seek.

Now the nature and quantization of the interestingness is in terms of how likely the residuals are under a particular model of the probability density function of the residual. As with standard EPP, we also sphere the data before applying the learning method to the sphered data.

2.2.3 Scatter Plot Matrix

This technique is one of the oldest and most popular methods for projecting high dimensional data (see, for example, Becker et al. [9] and Cleveland and McGill [26]). With this technique, instead of having only one representation of the data we can view the data from different perspectives. In the scatter plot matrix, which is also named the Draftman's display, we project the data onto every pair of axes. If we have an n-dimensional data set, the scatter plot matrix will have n rows and n columns and the i^{th} row and j^{th} column of this matrix is a plot of the data projected onto the i and j axes, showing the relations between each pair of variables and the nature of these relationships (direct or indirect, linear or nonlinear, degree of relationship, ...). The presence of outliers and clusters in the data can be identified as well. Figure 2.6 shows the scatter plot matrix for the well known iris data set (Ripley [135]) (a four dimensional data set).

The basic idea of the scatter plot matrix can be presented with a few variations:

1. The diagonal plot is simply a 45 degree line, so an alternative to that is to plot the univariate histogram (as in Figure 2.6), or to print the variable label.

2. Since the scatter plot matrix is symmetrical, it is possible to omit the plots below the diagonal, or use the upper portion to print the correlation coefficients.

3. It can be helpful to overlay some type of fitted curve on the scatter plots.

4. As proposed by Becker and Cleveland [8], we can use the scatter plot matrix with interaction techniques such as brushing and linking. Below we go a bit more deeply into these techniques.

The main drawback of the scatter plot matrix is that as we increase the dimensionality we leave less screen space for each projection. In [173], Ware and Beatty present a technique to lighten this problem by means of the use of colour.

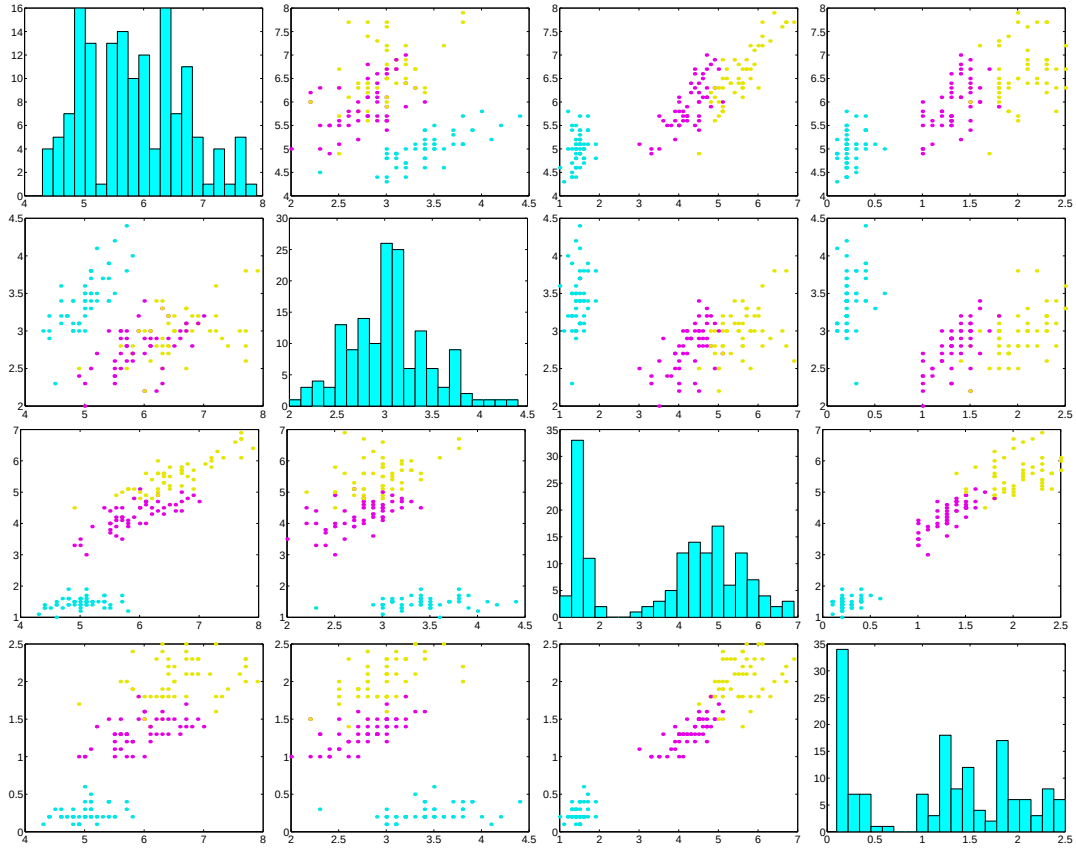


Figure 2.6: Scatterplot for the iris dataset

Each point is a coloured patch rather than a black point on a white background. We can use three of the dimensions to determine the amount of red, green and blue that gives the final colour.

Other visualization tools that use some of the ideas in the scatter plot matrix representation are the Hyperbox of Alpern and Carter [1] and the HyperSlice of Wijk and Leire [182]. The former presents the different orthogonal projections of the data set onto the *faces* of a subset of a wire frame model of an N-dimensional box, instead of onto the cells of a matrix grid. The latter uses the same matrix arrangement but presents slices of scalar functions of many variables instead of orthogonal data projections.

Brushing and Linking

The brushing technique has its origin as an interaction technique used in combination with scatter plot matrices. In [8], Becker and Cleveland state:

> The central object in brushing is the *brush*, a rectangle that is superimposed on the screen, ... The data analyst moves the brush to different positions on the scatter plot matrix by moving a mouse. There are four basic *brushing operations*—highlight, shadow highlight, delete, and label. Each operation is carried out in one of three *paint modes*—transient, lasting, and undo. At any time, the data analyst can stop the brushing and change one or more of the features—the shape of the brush, the operation, or the paint mode—and then resume the brushing. The brushing methodology provides a medium within which a data analyst can invent data analytic methods, which we call *brush techniques*.

Basically, with brushing operations we change some of the attributes (e.g. colour, glyph, visibility, labelling) of the points or lines we use to represent the data. The idea is to isolate clusters or other interesting subsets of a data set by, for example, painting that subset with a colour. Then the linking lets us identify the selected points in all the other projection panels in the scatter plot matrix, since the changed attribute changes at the same time in all views of the data. In this way, the highlight operation changes the colour or glyph of the points within the current panel in all the other panels, the shadow highlight changes the visibility in the other panels, the delete operation simply eliminates the points within the brush, and the label shows the associated label, if any, for the brushed points. In transient mode only the points under the brush are affected by the corresponding operation. In lasting mode, points inside the brush remain affected even after the brush no longer covers them. Finally, the undo mode is used to restore the attribute value changed by a previous brushing operation.

Although brushing and linking were proposed originally in the context of scatter plot matrices, these techniques can be used with any kind of graphical representation. With the use of the same attribute value for the data points in all representations, we can track coherent clusters or subsets of the data through different representations. With an animation we can follow the cluster of subsets of the data through the time evolution of the animation. We will make use of brushing and linking with the method proposed in Chapter 4.

2.3 Topology Preservation Methods

The techniques shown in the previous section are not adequate to reveal nonlinear structures, such as structures consisting of arbitrarily shaped clusters or curved manifolds, since they describe the data in terms of a linear subspace.

So, we need techniques that project the data in higher dimensions to lower dimensions (2 or 3 dimensions) without losing the characteristics of the local topology of the data. One way to achieve topology preservation is to preserve the distances between the points in the original data set, which means that:

1. Nearby data points give nearby projection points.
2. Distant data points give distant projection points.
3. If the projections of two data points are close, it is because, in the original high dimensional space, the two data points were close.
4. If the projections are distant, the original data points were distant.

Techniques such as multidimensional scaling achieve all of the previous properties. Techniques such as the self organizing map only achieve the second and third properties. In the following we give details of some of the topology preservation techniques.

2.3.1 Multidimensional Scaling

Every time we view a world map, we are seeing the result of applying a process of multidimensional scaling (MDS). Basically, in the world map we have a two-dimensional representation of the distance between cities that are on the surface of a sphere, and hence in a three-dimensional space.

The key idea of MDS is to give a visual representation of the distances between a set of points in a high-dimensional space. We want to maintain as much as possible the distance between the points in the high dimensional space[2]. To measure how closely we have reproduced the distances a *stress function* is used, whose general form is:

$$S = \sum_{ij} a_{ij}(\delta_{ij} - d_{ij})^2 \tag{2.22}$$

[2]Actually, MDS is not limited to the use of a distance matrix; it is possible to use any other kind of dissimilarity or similarity matrix (for example the matrix of correlations among variables).

where δ_{ij} is the distance between two data points $\mathbf{x}_i$ and $\mathbf{x}_j$, d_{ij} is the distance between the corresponding projected points $\mathbf{y}_i$ and $\mathbf{y}_j$, and a_{ij} is a scale or weighting factor. Taking

$$a_{ij} = \left(\sum_{ij} d_{ij}^2 \right)^{-1}$$

gives a classical stress function, known as Kruskal's stress [106]. We want to minimize the stress function. There exists multiple variants of MDS that use slightly different cost functions and optimization algorithms. In the following subsection we give details of one of these.

2.3.2 Sammon Mapping

Sammon mapping [140] performs MDS by using as a weighting factor

$$a_{ij} = \left(\delta_{ij} \sum_{ij} \delta_{ij} \right)^{-1}$$

that gives the stress function

$$S = \frac{1}{\sum_{ij} \delta_{ij}} \sum_{ij} \frac{(\delta_{ij} - d_{ij})^2}{\delta_{ij}} \tag{2.23}$$

which emphasizes the preservation of the local distances. One possible way to obtain a configuration of projected points that minimizes this stress function is to employ a gradient descent algorithm:

1. Initialize the Sammon mapping with random coordinates (2-d).
2. Calculate the relative pair wise error of each data point between spaces.
3. Calculate a gradient which shows the direction to minimize the error.
4. Move the data points in the Sammon mapping according to the gradient.
5. Repeat steps 2-4 until the error is below a given limit, or no improvement is seen.

This algorithm has some disadvantages. Firstly, it lacks generalization, which means that if we want to add new points we need to recalculate the projection. Secondly, a local minimum in the error surface could be reached, therefore we

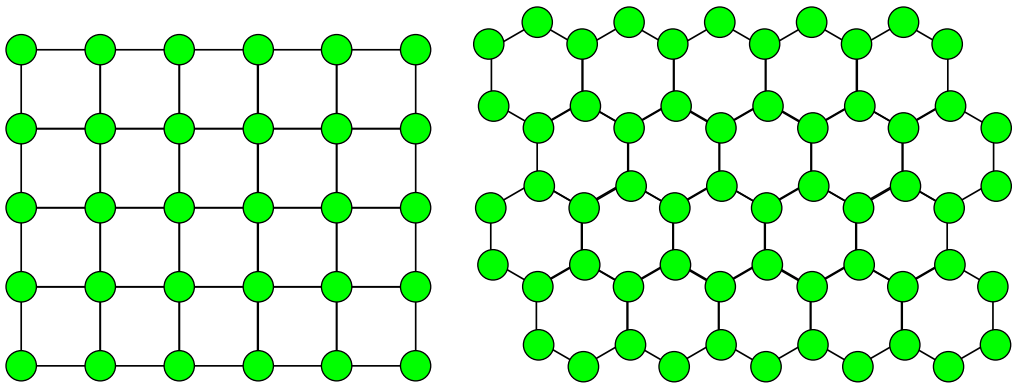
may need to try a significant number of experiments with different random initializations. Finally, it is very computationally intensive: with N points, in each iteration $N(N-1)/2$ distances must be calculated, resulting in $O(N^2)$ complexity per iteration.

However the problem of generalization can be alleviated using neural network implementations of the Sammon mapping; Ridder and Duin present in [31] a comparison.

2.3.3 Spring models

Within the class of MDS algorithms, one of the simpler are the spring model algorithms. These are based on the work of Fruchterman [49] in which the forces within a set of springs, that interconnect a set of rings, are simulated until an equilibrium is obtained. In the original work of Fruchterman, the rings were associated with nodes of a graph and the springs with the arcs. The equilibrium of the system gives a configuration with certain aesthetic characteristics (for example, the same length for all the arcs). In the context of MDS, it is considered that the springs are connecting every pair of points of a data set, the repulsion or attraction spring forces are proportional to the difference between the high dimensional distances and the bi-dimensional distances of the layout we want to obtain. The combination of forces applied over a point is used to calculate its velocity, and the velocity is used to obtain its position. At the equilibrium of the system, the projection maintains the similarities and dissimilarities of the high dimensional data set. Since it is necessary to calculate the forces for every pair of points and the number of iterations is proportional to N, the number of data points, the simulation gives an algorithm of complexity $O(N^3)$.

Chalmers presents in [20] a way to obtain a good approximation to the final configuration with less complexity. The key idea of the algorithm is the reduction of the amount of forces that it is necessary to calculate for each point. Every point has associated two sets. One is ordered and maintains the closest neighbours found so far. The second is re-calculated at every iteration by randomly sampling the full data set; if one of the points in the second set is closer than one of the points in the first, the former is substituted by the latter. In this way, at every iteration, the neighbour set will be more representative of the most similar points. For each point, only the forces applied by the springs associated with points in these two sets are considered. Since the size of these sets are constant, the



2D rectangular architecture 2D honeycomb architecture

Figure 2.7: Examples of SOM architectures.

complexity of the algorithm is reduced to $O(N^2)$ and the quality of the obtained layout is still good.

Morrison et al. [124, 125] introduce an additional improvement. In a first phase, only a set S of $\sqrt{N}$ points randomly sampled from the original data set are considered, and the full simulation is performed. The complexity of this step is $O(\sqrt{N}\sqrt{N})$, that is $O(N)$. Now, for every of the $N - \sqrt{N}$ remaining elements the closest point within the $\sqrt{N}$ points used in the first step is found; that closest point is called the 'parent' point. The point is located around the parent point taking also into account the forces of a sample of points over S. As a final stage of fine-tuning, a constant number of iterations of the simulation over the full data set are executed. The final complexity of the process is $O(N\sqrt{N})$.

A last improvement is proposed by Morrison and Chalmers [123]. This time the search for the parent elements is performed using distance discretization and a subset of selected points (pivots). The idea is to precalculate the distances to the pivots, and use those distances and the triangle inequality to reduce the distance calculations. With this strategy it is possible to reduce the complexity down to $O(N^{\frac{5}{4}})$.

2.3.4 Self Organizing Map

Contrary to the MDS methods, self organizing maps (SOMs) do not preserve distances, but these artificial neural networks are also used for dimensionality reduction, clustering and visualization. They were first described by Teuvo Kohonen [103], and so are also known as Kohonen maps.

The SOM has usually a rectangular or hexagonal two dimensional structure of interconnected artificial neurons (see Figure 2.7). The weights of these neurons can be seen as the coordinates of a vector that lies in the high dimensional space we want to explore (they are called model vectors, prototype vectors or centres). The process that organizes the positions of these vectors is an unsupervised competitive learning mechanism that works as follows:

1. Randomly select a training pattern, $\mathbf{x}$, from the input data set.

2. Find the neuron, c, whose centre is closest to the input pattern[3]; that neuron will be the *winning neuron* or the so called *Best Matching Unit* (BMU) for the pattern.

3. Adjust the centres toward the data vector for the winning neuron and all its neighbours using the following equation:

$$\Delta \mathbf{w}_i = \eta(\mathbf{x} - \mathbf{w}_i)\Lambda(i, c)$$

   where η is the learning rate, and Λ is often a monotonically decreasing function of the distance between i and c, known as the neighbourhood function. Normally this function is a Gaussian or a difference of Gaussians (though this is not monotonically decreasing).

4. Repeat the steps 1 to 3 for new inputs until some convergence criterion is reached.

This results in the network learning positions for the centres of its neurons which cover the input space and are determined by the density of the data in the input space.

The computational complexity of the SOM is $O(K^2)$, where K is the number of neurons in the grid: each learning step requires $O(K)$ computations, and to achieve a sufficient statistical accuracy the number of iterations should be at least some multiple of K. In some uses of the SOM, the number of neurons is of the order of the number of input samples; in these situations the computational complexity of the SOM is of the same order of magnitude as in the MDS algorithms.

[3]The closeness criterion is usually the Euclidean distance between the model vectors and the input pattern.

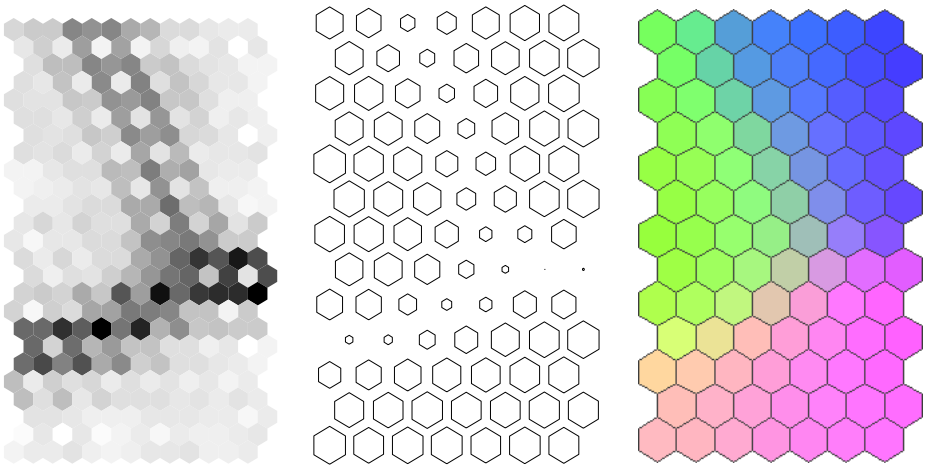


Figure 2.8: Examples of SOM visualisations methods.

Once trained, it is possible to use different methods to visualize the SOM structure (see Vesanto [170]). In Figure 2.8, we can see three different ways to represent a SOM trained using a data set with 3 clusters. The U-matrix (unified distance matrix), is perhaps the most popular method of displaying SOMs. The distance between adjacent neurons is calculated and presented with different colourings, darker colours representing larger distances. Light areas represent clusters and dark areas indicate cluster boundaries. The D-matrix can be constructed as an averaged version of the U-matrix, in which the size of each map unit is inversely proportional to the average distance to its neighbours. The similarity colouring, is obtained by spreading a colourmap on top of the principal component projection of the prototype vectors. Areas with similar colours are close to each other in the input space.

2.3.5 Other methods

All the visualization methods presented in this section can also be categorized within the field of *manifold learning*. That field is rich in techniques; the ones presented above are only some of them. Let us now briefly present some others.

Principal Curves

Principal curves are one of the nonlinear generalizations of principal components. They were first defined by Hastie and Stuetzle [77] as *self-consistent*, smooth,

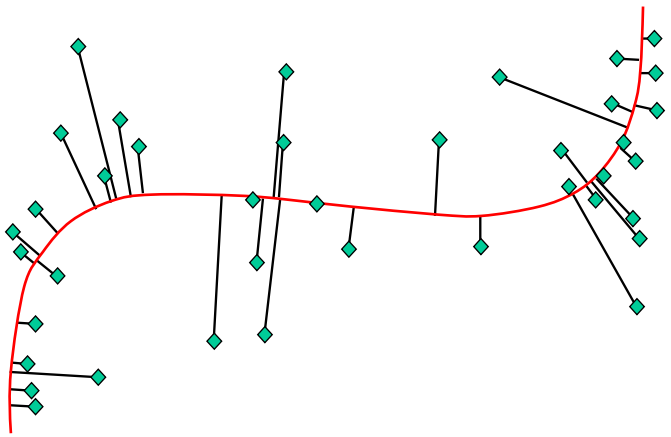


Figure 2.9: Example of principal curve through the middle of a bidimensional dataset.

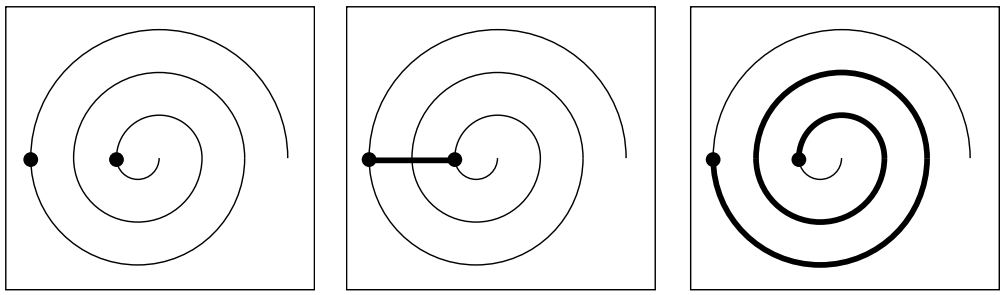
one-dimensional curves that pass through the *middle* of a p-dimensional data set, providing a nonlinear summary of the data (see Figure 2.9). An introduction and very good review of the field can be found at `http://www.iro.umontreal.ca/~kegl/research/pcurves/` (accessed on 20/July/2004).

The Elastic Net

Also known as the elastic map, it was first suggested by Durbin and Willshaw [39] as a solution to the travelling salesman problem, although Gorban and Zinevyev [73] have also applied it to visualization and manifold learning. It can be thought of as a variant of the SOM with a different learning rule. Recently a model has been proposed that unifies both paradigms [161, 162, 163].

Curvilinear Component Analysis

Previously known as "Vector Quantization & Projection" (VQP), but renamed "CCA" in order to evoke other *CA; it is a neural implementation of Sammon mapping preceded by a vector quantization to reduce the computational load (Demartines and Heráult [32, 34]). As with Sammon maps, it favours the local topology but the weighting does not use distance values in the high-dimensional space; instead CCA uses a decreasing function, F, of the low-dimensional distances. The function F is also parameterized by a neighbourhood width, λ, in



(a) (b) (c)

Figure 2.10: (a) two points in a spiral, (b) the Euclidian distance between the two same points and (c) the geodesic distance.

the style of the neighbourhood parameter of the SOM. The error function is:

$$E_{CCA} = \sum_{ij} (\delta_{ij} - d_{ij})^2 F_\lambda(d_{ij})$$

Isomap and Curvilinear Distance Analysis

The key idea in both algorithms is the use of an approximation to the geodesic distances instead of Euclidean distances. Every point is linked to the k closest points, or, alternatively, to the set of points closer than a radius ϵ. This gives a graph that is used to approximate the geodesic distances, first determining the shortest path between two points in the graph, then summing the Euclidean distances between the sequence of points in the path. Figure 2.10 clarifies the difference between the Euclidean distance and the geodesic distance.

In both algorithms, instead of using the full data set, a reduced set of points is selected. The differences between the two algorithms are in the way they select the subset of points and in the way they use the obtained geodesic distances. In the modern version of Isomap proposed by Tenenbaum et al. [160][4], a subset of points is obtained by random selection from the original data set. After computing the geodesic distances, the obtained matrix of distances is used to perform nonmetric MDS and to obtain the projection of the subset of points. In Curvilinear Distance Analysis [110] (CDA), the points in the graph are obtained through vector quantization of the original data set. Then CCA is applied over the subset using the geodesic distances. Finally, the projection of the original data points

[4]There exists a previous version of the algorithm, also by Tenenbaum [159], that used a different method to approximate the geodesic distances.

is computed using a piecewise linear interpolator. Lee et al. [111] have made a comparison between the two methods.

Isotop

Designed by Lee and Verleysen [112] to overcoming some of the limitation of the SOM, Isotop shares with CDA the initial vector quantization to obtain prototypes (or model vectors) of the original data set, the construction of a graph to approximate geodesic distances between these prototypes and the use of a piecewise linear interpolator in the last stage of the algorithm to obtain the projection of the original points by means of the projection of the prototypes. The main difference is how they obtain the projections of the prototypes. Now, instead of using CCA, an update rule similar to the SOM is used. However, the neighbourhood of the model vector is determined by the graph obtained in the first stages.

Locally Linear Embedding

Contrary to the metric MDS, the method of Locally Linear Embedding, proposed by Roweis and Saul [138], does not try to preserve distance but the local structure of the data. It assumes that the data manifold is locally linear and hence each data point can be obtained as a linear combination of its nearest neighbours. First, it calculates the weights, $\mathbf{W}_{ij}$, that best describe each point, $\mathbf{x}_i$ as a function of its neighbourhood, that is, the weights that minimize the error function:

$$\varepsilon(\mathbf{W}) = \sum_i \left\| \mathbf{x}_i - \sum_j \mathbf{W}_{ij}\mathbf{x}_j \right\|^2, \tag{2.24}$$

subject to $\sum_j \mathbf{W}_{ij} = 1$ and $\mathbf{W}_{ij} = 0$ if $\mathbf{x}_j$ does not belong to the set of neighbours of $\mathbf{x}_i$. Second, it finds the projections, $\mathbf{y}_i$ associated with each $\mathbf{x}_i$ that best reproduce the same reconstruction weights, that is, that minimize the following error function:

$$\phi(\mathbf{Y}) = \sum_i \left\| \mathbf{y}_i - \sum_j \mathbf{W}_{ij}\mathbf{y}_j \right\|^2, \tag{2.25}$$

Both problems can be solved using algebraic techniques.

A comparison with Isotop is presented in [109].

2.4 Representation of High Dimensional Data

In the previous sections, we have outlined some of the existing methods to project the high dimensional data onto a lower dimension that makes the graphical representation possible. In this section we present methods that, instead of searching for projections of the data, try to represent graphically all the dimensions of the data.

2.4.1 Chernoff faces

The original idea was presented by Chernoff in 1973 [23]. It consists of representing each high dimensional observation with a face cartoon. The multiple values of the observations are used in the drawing process as parameters which control features such as:

- the shape of the upper and lower part of the face (5 variables),
- the length of nose (1 variable),
- the vertical position, curvature and width of mouth (3 variables),
- the vertical position, separation, slant, eccentricity and size of eyes (5 variables),
- the position of pupils (1 variable),
- the vertical position, size and slant of eyebrows (3 variables).

Hence, different values gives different face cartoons (see Figure 2.11). The use of Chernoff's faces makes the visual identification of clusters easy and also facilitates the detection of outliers in high dimensional data. The explanation of this is that people have built-in face recognizers, or as Chernoff says [23]:

> People grow up studying and reacting to faces all of the time. Small and barely measurable differences are easily detected and evoke emotional reactions from long catalogue buried in the memory.

This method has been used in a broad range of disciplines: economics [122, 154], marketing [128], medicine [118], sociology [3, 141], and so on. One problem with this method is the subjective assignment of variables to features. Different assignments will give different face shapes. The effect of these assignments has

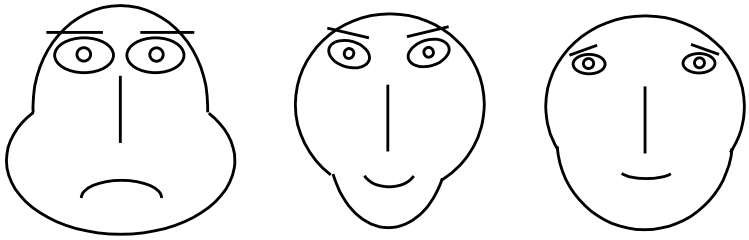


Figure 2.11: Three examples of Chernoff's faces.

been investigated by Chernoff and Rizvi [24]. As pointed out by Saxena and Navanneetham [143], the comparison with other methods discussed in [164, 142, 76], shows that Chernoff faces perform quite well in the identification of clusters. Saxena and Navanneetham themselves analyse the effect of the size, dimensionality and number of clusters when using Chernoff's faces. In [113], this visualization tool is evaluated in the context of binary data visualization.

Subsequent researchers have wanted to increase the dimensionality of the data set with which Chernoff faces can be used, by mean of using asymmetric faces [46]. Another variant uses, as graphical objects, schematic pictures of castles and trees [102]; or pictures which resemble bugs [25]. Taking advantage of the powerful graphical capabilities of the modern computer, more recently the construction of more realistic faces (not just caricatures) has been proposed [126, 117], and instead of doing a mapping from the data to face features, the data is used to change the emotional features of the faces; in this way, even an isolated face can transmit information to the viewer (for example, a happy face could mean a good financial situation).

2.4.2 Andrews' Curves

Andrews [2] described his curves in 1972, early on in the computing era; it is an interesting observation that he thought it necessary to counsel "an output device with relatively high precision ... is required". Current standard PC software is quite sufficient for the purpose. The method is another way to attempt to visualize and hence to find structure in high dimensional data. Each data point $\mathbf{x} = \{x_1, x_2, ..., x_d\}$ defines a finite Fourier series

$$f_{\mathbf{x}}(t) = x_1/\sqrt{2} + x_2 \sin(t) + x_3 \cos(t) + x_4 \sin(2t) + x_5 \cos(2t) + ... \qquad (2.26)$$

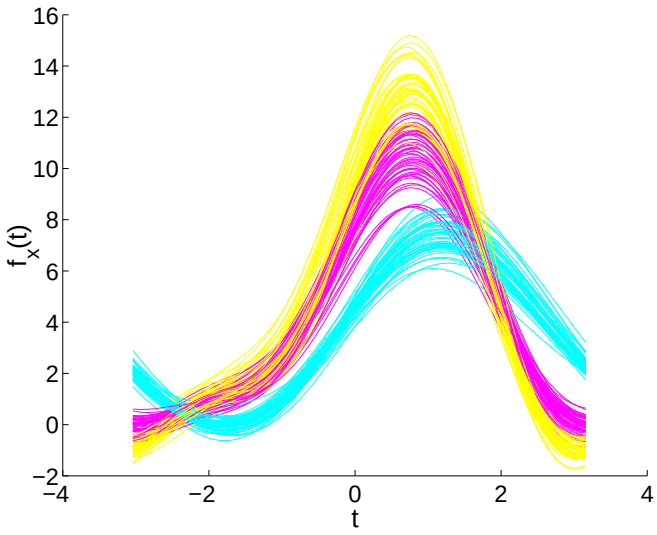


Figure 2.12: An Andrews' plot of the iris data set. It is clear that one type of iris is distinct from the other two but differentiating between the other two is less easy.

and this function is then plotted for $-\pi < t < \pi$. Thus each data point may be viewed as a line between $-\pi$ and π. This formula can be thought of as the projection of the data point onto the vector

$$\left(\frac{1}{\sqrt{2}}, \sin(t), \cos(t), \sin(2t), \cos(2t), \ldots\right)$$

If there is structure in the data, it may be visible in the Andrews' curves of the data. An example of Andrews' curves on the well known iris data set is shown in Figure 2.12. The data set is four dimensional (only the first four terms of (2.26) are used). We see that one type of iris is easily identifiable from the other two but there is some difficulty in differentiating between these two. Thus $t = 3$ gives us a value for a linear projection of the data which differentiates one type of data from the rest. But the fact that we cannot see any clear distinction between the other two types of iris in the figure does not necessarily mean that it is not possible to find a projection which will do so: the Andrews parameters are limited by the properties of the trigonometric functions (e.g. $\sin^2() + \cos^2() = 1$), which means that we have a very constrained relationship between the parameters of the projection for x_2 and x_3.

These curves have been utilized in fields as different as neurology [105], sociology [157], biology [127] and semiconductor manufacturing [134, 133]. Some of

their uses include the quality control of products [107], the detection of period and outliers in time series [44] or the visualization of learning in artificial neural networks [60]. Khattree and Naik [101] have suggested their utilization in robust design and in correspondence analysis.

Properties

These curves have several useful properties, some of which are:

1. Mean preservation. The function corresponding to the mean of a set of N multivariate observations, is the pointwise mean of the functions corresponding to these observations:

$$f_{\bar{\mathbf{x}}}(t) = \frac{1}{N}\sum_{i=1}^{N} f_{\mathbf{x}_i}(t)$$

2. Distance preservation. The distance between two functions is

$$||f_{\mathbf{x}}(t) - f_{\mathbf{y}}(t)||_{L_2} = \int_{-\pi}^{\pi} \big[f_{\mathbf{x}}(t) - f_{\mathbf{y}}(t)\big]^2 dt$$

   which is proportional to the Euclidean distance between the corresponding points since

$$||f_{\mathbf{x}}(t) - f_{\mathbf{y}}(t)||_{L_2} = \pi||\mathbf{x} - \mathbf{y}||^2 = \pi\sum_{i=1}^{d}(x_i - y_i)^2$$

3. One-dimensional projections. For a particular value of $t = t_0$, the function value $f_{\mathbf{x}}(t_0)$ is proportional to the length of the projection of the vector $(x_1, x_2, \cdots, x_d)$ on the vector

$$\mathbf{f_1}(t_0) = \big(1/\sqrt{2}, \sin(t_0), \cos(t_0), \sin(2t_0), \cos(2t_0), \cdots\big)$$

4. Linear relationships. If a point $\mathbf{y}$ lies on a line joining $\mathbf{x}$ and $\mathbf{z}$, then for all values of t, $f_{\mathbf{y}}(t)$ is between $f_{\mathbf{x}}(t)$ and $f_{\mathbf{z}}(t)$.

5. Also if the components of the data are uncorrelated with common variance σ^2, the Andrews' curves representations preserve that variance. This variance preservation property lets us perform a test of significance using the

curves, although, as noted by Khattree and Naik [101], this "*is less useful since most multivariate data are either correlated and/or have unequal variances across the variables*".

All these properties were noted by Andrews [2]. The last one was generalized by Goodchild and Vijayan [71] to the case of unequal and not necessarily orthogonal variances. Tests of significance at particular values of t are still possible, but not so the overall tests mentioned by Andrews.

However, the Andrews' curves have also a drawback, in that they suffer from strong dependence on the order of the variables, i.e. if we change the order of variables the shape of the curves is completely different. That is why Embrechts and Herbeg [42] propose to try different arrangements of the variables to find more suitable Andrews' curves. In Section 5.2, we propose a way of combining in the display two different arrangements. Besides, as pointed by Andrews [2], in the plots low frequencies are more readily seen than high frequencies. For this reason it is useful to associate the most important variables with low frequencies.

Variations

Some variations of the Andrews' curves have been proposed throughout the years. Andrews himself [2] proposed the use of different integers to give the general formulation:

$$f_{\mathbf{x}}(t) = x_1 \sin(n_1 t) + x_2 \cos(n_1 t) + x_3 \sin(n_2 t) + x_4 \cos(n_2 t) + \dots \qquad (2.27)$$

The restriction to using integers is because of the distance preserving property; without integers, this property is lost. Andrews compared the curve with values $n_1 = 2, n_2 = 4, n_3 = 8, \dots$ with the original formulation and concluded that the former is more space filling but more difficult to interpret when it is used for visual inspection.

Embrechts and Herzberg investigate in [42] the effect of re-scaling and re-ordering the coefficients and the interpretation of the plots when one or more coordinates are made equal to zero. They propose the use of other kinds of orthogonal functions such as Legendre and Chebychev polynomials. They give many examples of these variations using the iris data set. In [43], Embrechts completes this study with a new variation consisting of the use of wavelet functions. All these variations have been used later by Rietman and Layadi [133] as

a help to monitor the manufacture of silicon wafers; they also point to a previous work [134] in which Rietman used another variation consisting of drawing the Andrews' curves using polar coordinates.

A bivariate version of Andrews' plots has been proposed by Koziol and Werner [105]:

> Given two vectors of observations $x^T = (x_1, \cdots, x_p)$ and $y^T = (y_1, \cdots, y_p)$ where the $(x_i, y_i), i = 1, 2, \cdots, p$ are naturally paired, form the functions
>
> $$f_x(t) = x_1/\sqrt{2} + x_2 \sin(t) + x_3 \cos(t) + x_4 \sin(2t) + x_5 \cos(2t) + \cdots$$
> $$f_y(t) = y_1/\sqrt{2} + y_2 \sin(t) + y_3 \cos(t) + y_4 \sin(2t) + y_5 \cos(2t) + \cdots$$
>
> and plot $(t, f_x(t), f_y(t))$ for a set of t-values in the range $-\pi \leq t \leq \pi$.

They use this variation in an investigation about the neurological consequences of cerebral artery occlusion.

A similar idea to the previous one, but this time to obtain a three dimensional Andrews' plot, has been proposed by Wegman and Shen [179]. They were concerned with the connection between Andrews' curves and the grand tour[5] noted by Crawford and Fall [29]. They show that Andrews' curves are not a real one-dimensional grand tour. The problem is that Andrews' curves do not exhaust all possible orientations of a one-dimensional vector. They propose a generalization of Andrews' curves which is more space filling and which can be used to obtain a bi-dimensional pseudo grand tour. Now, two orthogonal vectors are used:

$$\begin{aligned} \mathbf{w}_1 &= \sqrt{\frac{2}{d}}\Big(\sin(\lambda_1 t), \cos(\lambda_1 t), ..., \sin(\lambda_{\frac{d}{2}} t), \cos(\lambda_{\frac{d}{2}} t)\Big) \\ \mathbf{w}_2 &= \sqrt{\frac{2}{d}}\Big(\cos(\lambda_1 t), -\sin(\lambda_1 t), ..., \cos(\lambda_{\frac{d}{2}} t), -\sin(\lambda_{\frac{d}{2}} t)\Big) \end{aligned}$$

with the λ_j linearly independent over the rationals to increase the space filling property of the curves, but losing the distance preservation property[6] (we will come back to this variation in Chapter 4).

[5]The grand tour [4] is a multivariate visualization method that consists of looking at the data from all points of view by presenting a continuous sequence of low dimensional projections; we will discuss this, in more detail in Section 2.5.

[6]Khattree and Naik [101] point to Gnanadesikan [70] who attribute a special case of this formulation to Tukey. Tukey used as lambdas $1, \sqrt{2}, \sqrt{3}, \sqrt{5}, \ldots$.

More recently, Khattree and Naik [101] have suggested the function:

$$g_{\mathbf{y}}(t) = \frac{1}{\sqrt{2}}\Big\{y_1 + y_2\big(\sin(t) + \cos(t)\big) + y_3\big(\sin(t) - \cos(t)\big) + y_4\big(\sin(2t) + \cos(2t)\big) + y_5\big(\sin(2t) - \cos(2t)\big) + \cdots\Big\}, \quad -\pi \leq t \leq \pi \tag{2.28}$$

So, every y_i is exposed to a sine function as well as a cosine function. As they note, one of the advantages of this formulation is that the trigonometric terms in (2.28) do not simultaneously vanish at any given t. They also establish an interesting relation between the Andrews' curves and the eigenvectors of a symmetric positive definite circular covariance matrix.

2.4.3 Parallel Coordinates

Parallel coordinates are an invention of Alfred Inselberg originally developed as a device for visualizing multidimensional geometry [86]; it was their utility as air traffic collision detectors [88] which initially brought serious and broader interest to this new tool [85]. Later, Wegman proposed their use as a data analysis tool [174] and enhanced them by combining them with the grand tour [175].

The parallel coordinate plots can be thought of as a generalization of the two-dimensional Cartesian plot. Instead of using orthogonal axes, the axes are drawn parallel to each other. Instead of using a "dot" to represent the location of a "point" and the values of its coordinates, a "line" is used which connects the coordinates of the point on the axes. So the points became lines (or, to be more precise, polylines) as we can see in Figure 2.13[7]. If we draw the lines associated with points lying on the same line, we discover that they intersect at a point, which is the dual of the line where the points lie. So, in parallel coordinates plots, the dual of points are lines and the dual of lines are points (see Figure 2.14). Also, now we can draw as many axes as we want, so we can represent points of dimensionality greater than three. In Figure 2.15 an interval of points on a line in a ten dimensional space is represented. The set of dualities are:

- A point in Cartesian coordinates becomes a line in parallel coordinates (a poly line if we consider more than two dimensions, and hence we draw more than two parallel axes) and vice versa.

[7]Figures 2.13 to 2.16 are reproduced from [87].

- An ellipse in Cartesian coordinates maps into a "*line hyperbola*" in parallel coordinates. Maybe the Figure 2.16 can clarify the strange notion of line hyperbola. In general a point conic in Cartesian coordinates becomes a line conic in parallel coordinates.

- Rotations in Cartesian coordinates become translations in parallel coordinates and vice versa.

- Points of inflection in Cartesian space become cusps in parallel coordinate space and vice versa.

Because of these duality properties, parallel coordinate displays allow interpretations of statistical data in an analogous way to the two-dimensional Cartesian scatter plots. In the statistical setting, as pointed out by Wegman [175], the following interpretations can be made:

- For highly negatively correlated pairs, the dual line segments in parallel coordinates tend to cross near a single point between the two parallel coordinates axes. So an X-shape between axes signals negative correlation between the respective variables.

- For highly positively correlated data, lines tend not to intersect between the parallel coordinates axes. Parallel or almost parallel lines between axes indicates positive correlation between variables.

Two of the most common objections to parallel coordinate plots are the dependency on the order of the axes in order to identify the relations between variables (the pairwise comparison between consecutive axes is easy, but this comparison is not so easy when the axes are not adjacent) and the bad *data-ink ratio*[8] or heavy overplotting (as lines are used to represent points, when the size of the data set is too large, the parallel coordinate plot becomes a messy clutter of ink). To minimize the first, Wegman presents in Appendix A of [174], an algorithm for constructing a minimal set of permutations to insure adjacency of every pair of axes. If the number of axes is d, only $(d+1)/2$ permutations are needed. To address the second, in the same paper, Wegman proposes the utilization of parallel coordinate density plots, that is, the use of colours to represent the density of observations. More details were given in a subsequent paper [176].

[8]This concept is discussed by Tufte in [166].

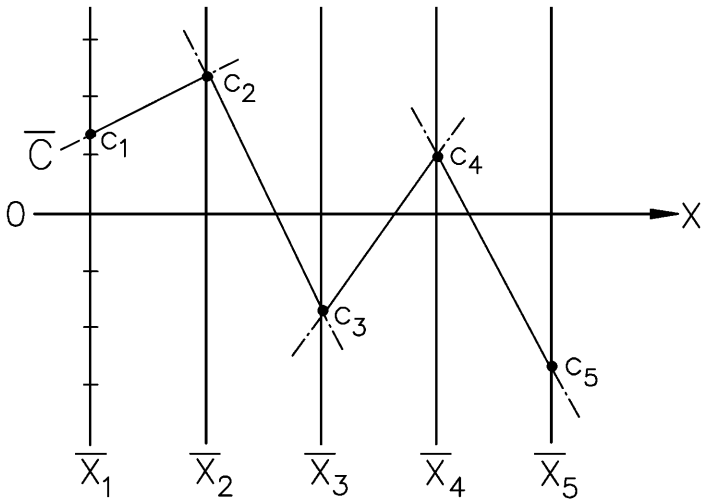


Figure 2.13: Polygonal line $\overline{C}$ represents the point $C = (c_1, c_2, c_3, c_4, c_5)$.

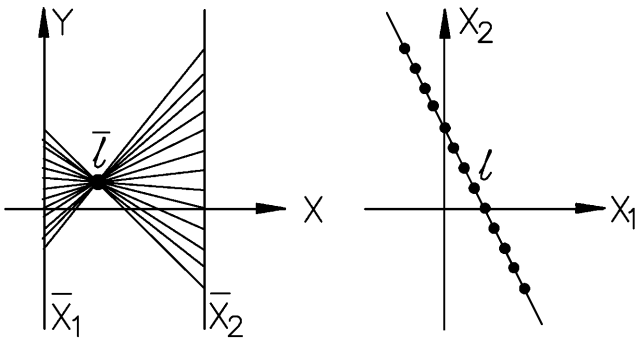


Figure 2.14: The dual of the line ℓ is the point $\bar{\ell}$.

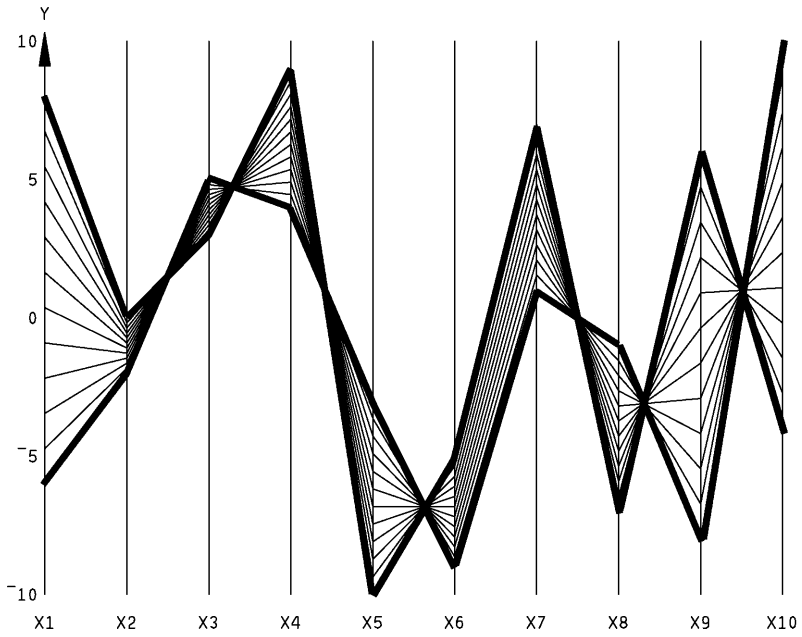


Figure 2.15: Interval of points on a line in a ten-dimensional space. Darker polygonal lines represent the end-points. The parallelism of the segments of line between the axes X6 and X7 is an indication of the collinearity of the points in the X6-X7 plane. If the pattern of parallelism had appeared also between, let us say, the axes X5 and X6, we could conclude that the points were coplanar in the X5-X6-X7 three-dimensional space.

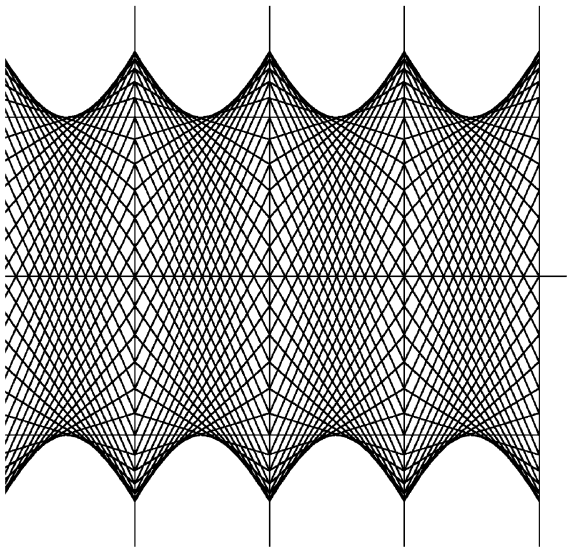


Figure 2.16: Parallel coordinate plot of a five-dimensional hyper-ellipse. Note the hyperbolic envelopes (the envelope of a line conic is a point conic).

The use of parallel coordinates as a data analysis tool has been perfected later by means of using the grand tour and the saturation brushing mechanism [175, 177]. This last technique allows a very fast calculation of the densities of the lines. Quoting Wegman and Solka [180]:

> The idea of saturation brushing is to desaturate a brushing colour until it contains only a very small component of colour and hence is very nearly black. Most modern computers have a so-called α-channel which allows for compositing of overplots. The α-channel is used in computer graphics as a device for incorporating transparency. However, by using such a device to build up colour intensity, we can obtain a visual indication of how much overplotting there is at a pixel. In effect, the brighter, more saturated a pixel is, the more overplotting.

An alternative to saturation brushing for processing high volume data sets is the hierarchical parallel coordinates proposed by Fua et al. [50]. They "*use data aggregation techniques to collapse data into clusters, and show the population and extents of clusters with bands of varying translucency*". Other improvements have to do with interaction techniques [152, 78, 153, 74].

2.5 Grand Tour

The idea of the grand tour method, introduced by Asimov [4] and Buja and Asimov [18], is to generate a continuous sequence of low dimensional projections of a high dimensional data set. The animation obtained can be thought of as a generalization of rotations in high-dimensional space. From that point of view, the grand tour shares a common objective with exploratory projection pursuit techniques. In both cases the human ability for visual pattern recognition is exploited.

To create a two dimensional grand tour we need to generate a sequence of planes in the high dimensional space of the data. Two conditions are required: (i) the sequence should be dense in the set of all planes in the high dimensional space, (ii) the sequence should be smooth to give a visual impression of the data points moving in a continuous way. Several algorithms have been proposed to achieve these two conditions (see Asimov [4], Asimov and Buja [5] and Wegman and Solka [180] for a deeper treatment). They are based on obtaining a general rotation in the high dimensional space, the dimension of which is d. To obtain a

$$R_{ij}(\theta_{ij}) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cos(\theta_{ij}) & \cdots & -\sin(\theta_{ij}) & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & \sin(\theta_{ij}) & \cdots & \cos(\theta_{ij}) & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

(i^{th} column, j^{th} column, i^{th} row, j^{th} row)

Figure 2.17: Rotation matrix for the $\mathbf{e}_i\mathbf{e}_j$ plane through an angle of θ_{ij}.

frame of the grand tour we multiply the data set by a generalized rotation matrix $\mathbf{Q}$ of dimension $d \times d$. The matrix $\mathbf{Q}$ is obtained by multiplying $p = d(d-1)/2$ matrices, $R_{ij}(\theta_{ij})$, each of which rotates the $\mathbf{e}_i\mathbf{e}_j$ plane through an angle of θ_{ij} (an outline of this matrix is shown in Figure 2.17). Now the algorithms differ in the way they obtain the angles θ_{ij}. In *the Asimov-Buja Winding Algorithm* the p angles θ_{ij} are taken from the coordinates of the vector $\alpha(t) = (\lambda_1 t, \lambda_2 t, \ldots \lambda_p t)$ that defines a mapping from $\mathbb{R}$ onto $[0, 2\pi]^p$, where t is the time parameter, the $\lambda_j t$ are taken modulo 2π and $\lambda_1, \ldots, \lambda_p$ are real numbers linearly independent over the rational numbers. With these conditions this vector defines a curve over a p-torus that does not intersect itself. *The Random Curve Algorithm* just randomly takes two points $\mathbf{s}_i, \mathbf{s}_j \in [0, 2\pi]^p$ and creates a linear interpolant between them going from $\mathbf{s}_i$ to $\mathbf{s}_j$, then takes a third point $\mathbf{s}_k$ and join it with $\mathbf{s}_j$ and so on. In *The Fractal Curve Algorithm* a p-dimensional Hilbert or Peano curve [139] is constructed that tours through the p-dimensional hypercube $[0, 2\pi]^p$.

As we have commented in Section 2.4.2, although it does not strictly obtain a grand tour, the much simpler of the algorithms is based on the use of the variant of Andrews' curves proposed by Wegman [179].

Originally, the grand tour was designed with the aim of obtaining two dimensional projections; with the arrival of parallel coordinates it is possible to use higher dimensional projections, which was noted first by Edward J. Wegman in [179]. Wegman and Luo [177] give an interesting example about the combined use of parallel coordinates and grand tours.

An enhancement of the grand tour, called the Tracking Grand Tour (TGT), is presented by Huh and Kim in [83]. In the TGT, during the grand tour, the old

frames instead of being substituted by the new ones, are maintained for a while, "*that shows the trace of the touring process as small 'comet trails' of the projected points*".

In [178], Wegman et al. propose another variation of the grand tour, named the Image Grand Tour. This time the idea of the grand tour is used to combine in a single gray scale image, several multispectral images of the same scene. The grand tour is through the different combinations of the multispectral images; a continuously changing gray scale image is obtained with variable contributions from the collected images. The image grand tour has been successfully applied to the detection of mines in a minefield. Symanzik et al. [158] show more applications of the image grand tour.

2.6 Software tools

For most of the methods outlined in this chapter, there exist tools that automate the process of obtaining the graphical representation. Here we give a list with the web address from which they can be downloaded and a short comment about their visualization capabilities (for some of them it is possible to find a more complete review in [6]).

- XmdvTool (`http://davis.wpi.edu/~xmdv/`): scatter plots, star glyphs, hierarchical parallel coordinates, dimensional stacking, proximity colouring.
- GGobi (`http://www.ggobi.org/`): parallel coordinates, grand tour, brushing and linking, scatter plots.
- Parallel Coordinate Explorer (`http://www.cs.uta.fi/~hs/pce/`): parallel coordinates and brushing.
- CrystalVision (`ftp://www.galaxy.gmu.edu/pub/software/CrystalVision.exe`): parallel coordinates, grand tour, saturation brushing.
- parvis (`http://home.subnet.at/flo/mv/parvis/`): parallel coordinates, axis histograms, brush fuzziness.
- ViDaExpert (`http://www.ihes.fr/~zinovyev/vida/vidaexpert.htm`): linear discriminant analysis, PCA, elastic maps.
- HiSee (`http://hisee.sourceforge.net/about.html`): PCA, Sammon mapping.

- Visual Attribute Explorer (`http://www.alphaworks.ibm.com/tech/visualexplorer`): parallel coordinate plots, attribute bar chart.
- Martin's Parallel Coordinate Curves Applet (`http://www.dcs.napier.ac.uk/~marting/parCoord/GenParCoordApplet.html`): parallel coordinates curves.
- VisDB (`http://www.dbs.informatik.uni-muenchen.de/dbs/projekt/visdb/visdb.html`): pixel-oriented techniques, stick figure icons, parallel coordinates.
- SOM Toolbox for Matlab (`http://www.cis.hut.fi/projects/somtoolbox/`): self-organizing maps, Sammon mapping, CCA, PCA.
- Datatool (`http://www.datatool.com/`): scatter plots, colour scatter plots.
- Data Loom (`http://s92417348.onlinehome.us/software/dataloom/`): parallel coordinates (for Macintosh).
- The Visualization Toolkit (`http://public.kitware.com/VTK/index.php`): C++ library to make the creation of useful graphics and visualization applications much easier.
- Mondrian (`http://www1.math.uni-augsburg.de/Mondrian/`): parallel coordinates, saturation brushing, mosaic plot, scatter plots, maps, bar charts, histograms and Box plots.
- Manet (`http://www1.math.uni-augsburg.de/Manet/`): scatter plots, box plots, mosaic plots, histograms, polygon plots.
- Andrews Plots add-in for Microsoft Excel (`http://www.herts.ac.uk/business/staff_public/nhspencer_public/research/andrewsplots/index.htm`): Andrews' plots.
- Matlab package for Isomap (`http://isomap.stanford.edu/IsomapR1.tar`): Isomap projections.
- Matlab code for LLE (`http://www.cs.toronto.edu/~roweis/lle/code.html`): LLE projections.
- John Aldo Lee code for Non-Linear Projections (`http://www.dice.ucl.ac.be/~lee/`): self-organizing maps, Sammon mapping, CDA, Isotop.

2.7 Conclusions

In this chapter we have shown how wide and diverse are the data mining and visualization fields. This fact is even clearer if we consider the lack of a commonly accepted taxonomy for the field. The variety of tools, far from being a drawback, is an advantage, or actually a need. Some of them are more suitable for identifying clusters, others can handle more data records. For each data set there exists an appropriate tool. The main conclusion we can state here is that there is not a tool better than the others over all data sets. The important thing is to provide data analysts with a set of tools, in which they can choose the tool most suitable for the data set they are investigating.

Chapter 3

EPP Neural Networks

In this chapter we review the two neural methods of performing Exploratory Projection Pursuit (EPP) introduced in Section 2.2.2: the Output Functions neural network and the Maximum Likelihood neural network. We begin by comparing their performance in the identification of leptokurtotic and bimodal directions on an artificial data set and show that, though the Output Functions EPP algorithm takes rather longer to converge, it is more accurate than the Maximum Likelihood EPP method. This suggests combining the best features of both. We show that the combined algorithm has fast and accurate convergence on an artificial data set and no less informative projections of a real data set.

The process for comparison of the EPP neural networks using real data sets presented in this Chapter is based on a novel use of cluster validation indices and gives a framework for comparison that can be used for other EPP algorithms.

We also suggest a new algorithm for the identification of skewed directions using the Maximum Likelihood neural network, and we compare this with the results obtained when the Output Function neural network is used to perform the same identification.

Finally, we investigate a way to improve the convergence of the three EPP algorithms. We do so with a specific initialisation of weights instead of using a random initialisation. To get the initial values we use a graphical representation for high dimensional data: the Andrews' curves.

PCA network:	
• feedforward:	$y_i = \sum_{j=1}^{N} w_{ij} x_j, \ \forall i$
• feedback:	$e_j = x_j - \sum_{i=1}^{M} w_{ij} y_i, \ \forall j$
• weight update:	$\Delta w_{ij} = \eta y_i e_j$
Output function EPP network:	
• feedforward:	$y_i = \sum_{j=1}^{N} w_{ij} x_j, \ \forall i$
• feedback:	$e_j = x_j - \sum_{i=1}^{M} w_{ij} y_i, \ \forall j$
• weight update:	$\Delta w_{ij} = \eta f(y_i) e_j$
	$f(y) = \begin{cases} y^3 \text{ or } y - \tanh(y) \text{ to identify leptokurtosis} \\ -y^3 \text{ or } \tanh(y) \text{ to identify bimodality} \end{cases}$
Maximum likelihood EPP network:	
• feedforward:	$y_i = \sum_{j=1}^{N} w_{ij} x_j, \ \forall i$
• feedback:	$e_j = x_j - \sum_{i=1}^{M} w_{ij} y_i, \ \forall j$
• weight update:	$\Delta w_{ij} = \eta y_i \operatorname{sign}(e_j) \lvert e_j \rvert^{p-1}$
	$p = \begin{cases} 3 \text{ to identify leptokurtosis} \\ 1 \text{ to identify bimodality} \end{cases}$

Table 3.1: The PCA and EPP neural networks.

3.1 A comparison using an artificial data set

Since we wish to compare two neural networks (see Table 3.1), we create artificial data sets in which we know the nature of the structure in the data set and for which we know where the structure lies. Thus we can measure how quickly the networks converge to the correct solutions.

3.1.1 Output Function EPP neural network

First we create 5 dimensional data such that 4 dimensions contain data drawn iid (independent identically distributed) from zero mean and different variance[1] Gaussian distributions, while the fifth dimension contains kurtotic data: we draw this also from a Gaussian distribution but randomly (in 20% of cases) substitute the sample with a small random number drawn from a uniform distribution between -0.005 and $+0.005$. This gives a sharp peak in the distribution round 0 and so creates kurtosis.

We use the Output Function EPP neural network (see Table 3.1) with $f(y) =$

[1]The actual values of the variances were 1,2,3,4,5, although that does no affect the final experiment due to the sphering preprocess (without reduction of dimensionality).

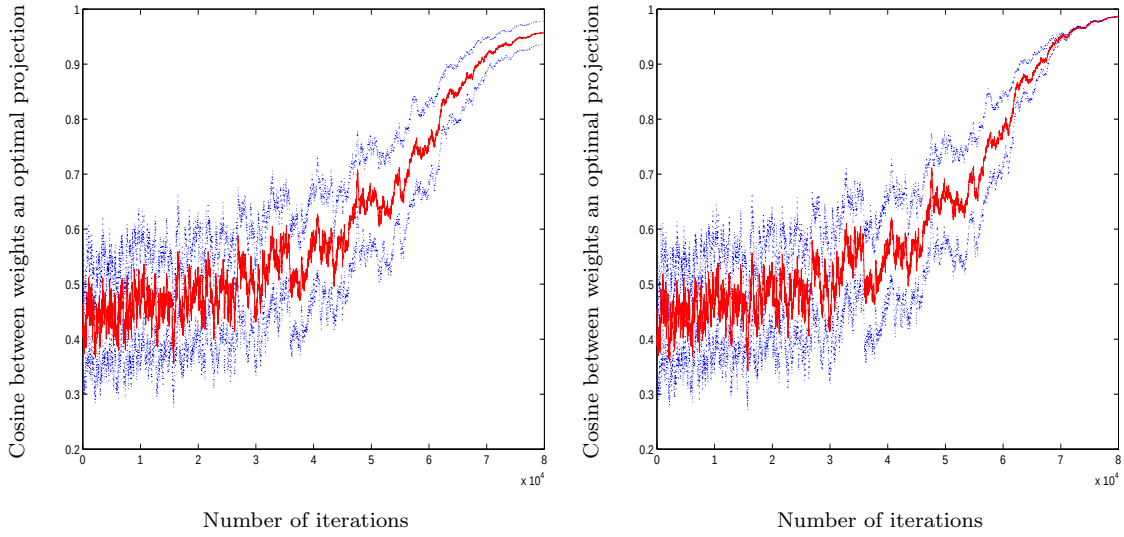


Figure 3.1: *Left*: convergence of 67 experiments towards the optimal (kurtotic) direction with $f(y) = y^3$. The graph shows the mean value of the cosine of the angle between the network weights and the optimal weight at each iteration and one standard deviation either side of the mean. *Right*: convergence of the best 64 experiments.

y^3 and repeat the experiment 100 times with different initial random weights. The initial learning rate was $\eta = 0.001$ which was linearly annealed to zero during the course of the training. It has been shown [55] that this algorithm causes the weights to converge to identify the higher order structure in a data set. The cubic nature of the function used makes the learning rule inherently unstable and, in 33 cases we found overflow. The convergence of the other 67 cases are shown in the left part of Figure 3.1 in which we show the mean value of the cosine of the angle between the network weights and the optimal value at each iteration (solid line) and one standard deviation above and below this figure. In 3 experiments, the network did not approach convergence after 80000 iterations. The results of the remaining 64 cases are shown in the right part of Figure 3.1.

However, we might expect that a learning rule which had a cubic element might be unstable; we have, therefore, used a function whose Taylor series expansion is dominated by the cubic term for sphered data (see [57]). Thus in Figure 3.2, we show the convergence of 100 simulations on the same type of data (left figure) and then the results of the best 90 experiments in the right figure when we use $f(y) = y - \tanh(y)$ (whose Taylor expansion starts: $(1/3)y^3 - (2/15)y^5 + (17/315)y^7 \ldots$). We see that we not only improve stability but also improve the speed of convergence with this function.

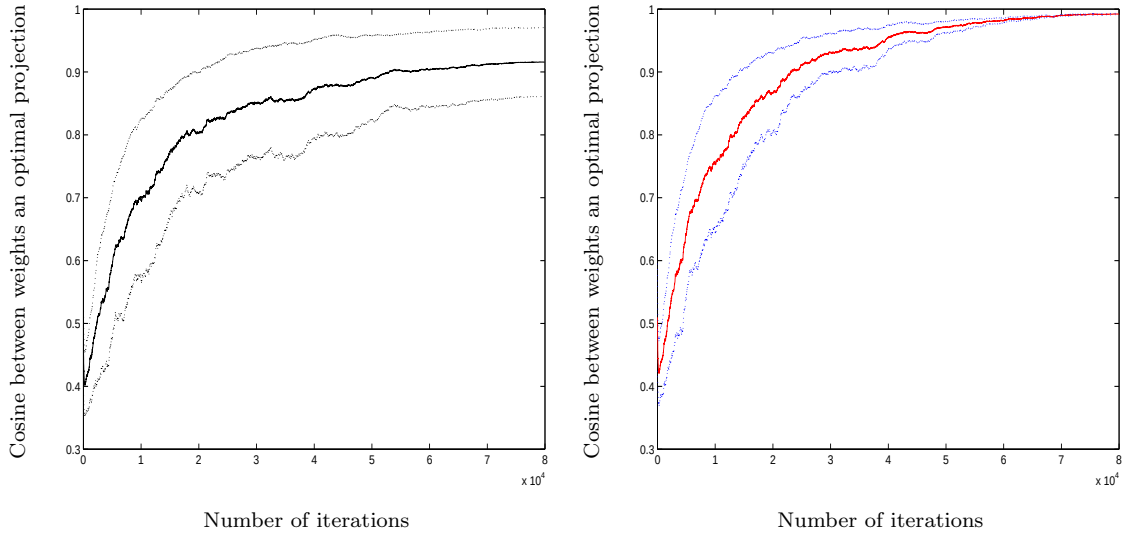


Figure 3.2: *Left*: convergence of 100 experiments towards the optimal (kurtotic) direction. The graph shows the mean value of the cosine of the angle between the network weights and the optimal weight at each iteration and one standard deviation either side of the mean. *Right*: convergence of the best 90 experiments. These graphs show results when using $f(y) = y - \tanh(y)$.

Similarly, we may create data which has one platykurtotic dimension by sampling from a zero mean Gaussian distribution and then randomly adding 5 to the sample or subtracting 5 from the sample. This gives a bimodal distribution which the function $f(y) = \tanh(y)$ (with Taylor expansion: $y - (1/3)y^3 + (2/15)y^5 - \ldots$) can be used to find. Thus in Figure 3.3, we show the results of 100 experiments searching for this dimension in the left figure and the best 99 (only one had not converged by iteration 80000) in the right of the figure.

3.1.2 The Maximum Likelihood EPP neural network

We construct the same type of data sets as before and use the Maximum Likelihood rules with $p = 3$ to search for leptokurtosis and $p = 1$ for the platykurtotic bimodal data set. We see in Figure 3.4 that convergence is not quite as accurate with this method but that it seems more sure: we have no failures to report on this data set with this method.

3.1.3 A Combined Algorithm

The two EPP algorithms were derived from different perspectives and each used changes to a Principal Component Analysis neural network *in different parts of*

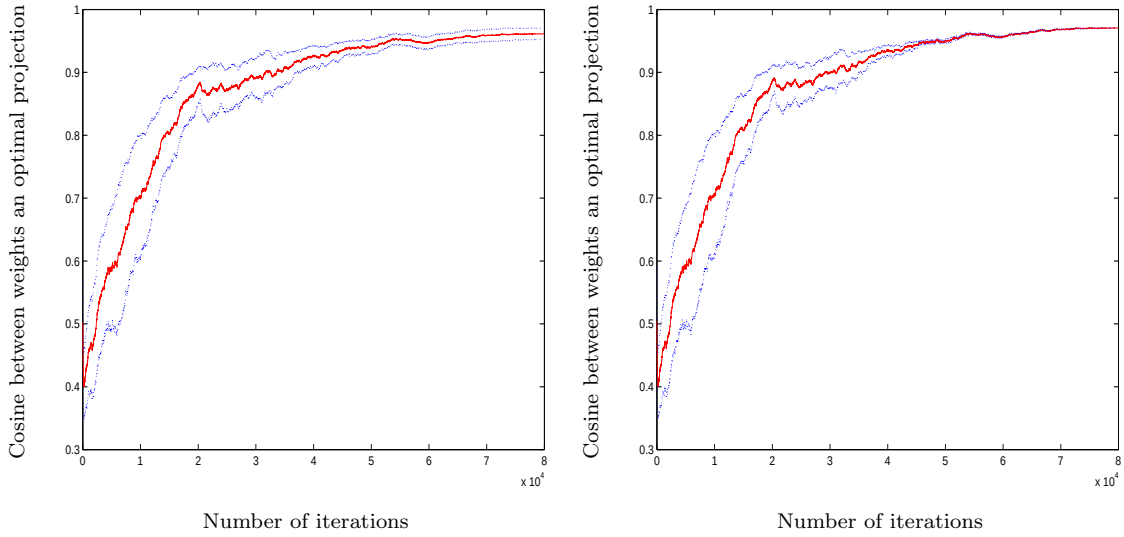


Figure 3.3: *Left*: convergence of 100 experiments to optimal bimodal dimension when using tanh() in the learning rule. *Right*: convergence of the best 99 simulations.

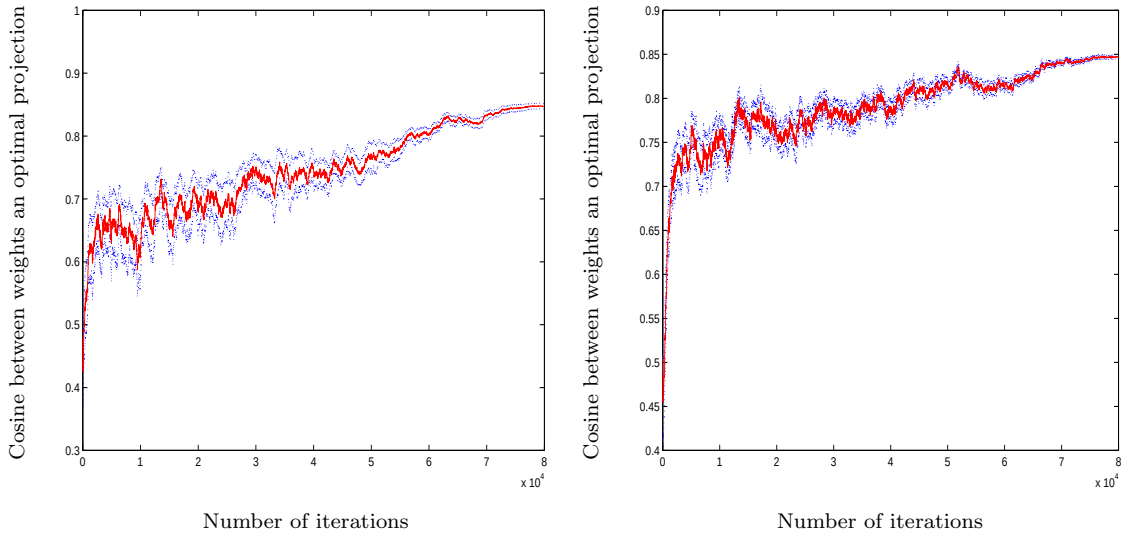


Figure 3.4: The convergence of the Maximum Likelihood Learning rules. *Left*: using $p = 3$ on leptokurtotic data. *Right*: using $p = 1$ on the bimodal platykurtotic data.

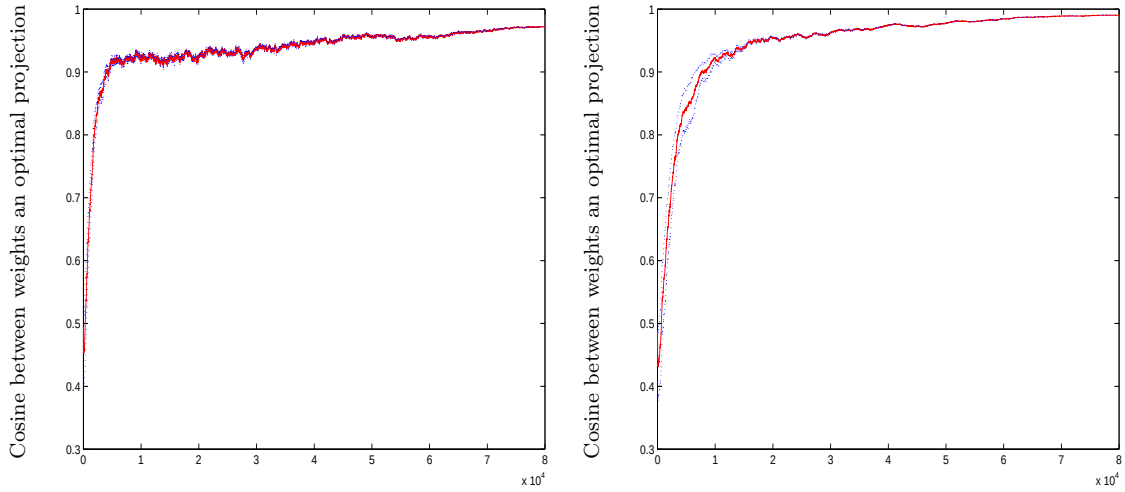


Figure 3.5: Convergence of the algorithm using the combined learning rule. *Left*: bimodal data. *Right*: leptokurtotic data

the weight change algorithm. This might suggest that we can get the best of both worlds by combining these algorithms. Figure 3.5 shows the convergence of a new algorithm which uses:

Feedforward:

$$y_i = \sum_{j=1}^{N} w_{ij} x_j, \forall i$$

Feedback:

$$e_j = x_j - \sum_{i=1}^{M} w_{ij} y_i$$

Weight change:

$$\Delta w_{ij} = \eta f(y_i)\,\mathrm{sign}(e_j)|e_j|^{p-1}$$

where:

- $f(y)$ is the $\tanh(y)$ function and $p = 1$ for the bimodal data.
- $f(y) = y - \tanh(y)$ and $p = 3$ for the leptokurtotic data.

We see a very fast, accurate and robust convergence to the optimal weights.

3.2 A comparison using a real dataset

In the previous section we have shown that the new EPP neural network, in spite of not being derived from a minimization of error criterion (as were the Output Function and Maximum Likelihood neural networks), performs very well on artificial data sets. In this section we use a real data set and several indices as a more objective way to make the comparison.

The use of artificial data sets to compare EPP algorithms is convenient since we know where the structure lies, so we can confirm that the convergence is appropriate. When real data sets are used we must rely on our subjective judgement about which projection is the best. Besides, we are limited by the number of such projections we can evaluate. With cluster validation techniques, different kinds of objective indices exist to determine the correct number of clusters in the analyzed data set [14]. We show that such indices can also be used in the comparison of EPP algorithms to avoid the forementioned drawbacks. Now the problem is not the determination of the number of clusters, which is a priori known, but the evaluation of the goodness of the projections obtained by an EPP algorithm.

3.2.1 Validation indices

Here we introduce the indices: C [82], the *Goodman-Kruskal* [72], the *Silhouette* [137], the *Dunn's* [38], and the *Davies-Bouldin* [30] indices, that we have used in the comparison of the EPP neural networks.

C Index

Let SD be the sum of distances over all pairs of samples from the same cluster, p be the number of those pairs (if we have n clusters with sizes, $cs_1, cs_2, \ldots, cs_n$, then $p = cs_1(cs_1 - 1)/2 + cs_2(cs_2 - 1)/2 + \ldots + cs_n(cs_n - 1)/2$), SD_{min} the sum of the p smallest distances if all pairs of samples are considered (even samples from different clusters), and SD_{max} the sum of the p largest distances between all pairs. The C-Index is defined as:

$$C = \frac{SD - SD_{min}}{SD_{max} - SD_{min}}$$

A small value of C indicates a good clustering. In the context of the experiments described in this chapter, this means that a good projection has been obtained.

Goodman-Kruskal Index

For this index all possible quadruples (q, r, s, t) of input samples are considered. Let $d(x, y)$ be the distance between the samples x and y. A quadruple is called *concordant* if one of the following two conditions is true:

- $d(q, r) < d(s, t)$, q and r are in the same cluster, and s and t are in different clusters.
- $d(q, r) > d(s, t)$, q and r are in different clusters, and s and t are in the same cluster.

A quadruple is called *disconcordant* if one of following two conditions is true:

- $d(q, r) < d(s, t)$, q and r are in different clusters, and s and t are in the same cluster.
- $d(q, r) > d(s, t)$, q and r are in the same cluster, and s and t are in different clusters.

Let Q_c and Q_d denote the number of concordant and disconcordant quadruples, respectively (note that some quadruples may be neither concordant nor disconcordant). Then the Goodman-Kruskal index is defined as:

$$GK = \frac{Q_c - Q_d}{Q_c + Q_d}$$

Large values of GK, that is many concordant and few disconcordant quadruples, indicate a good clustering and hence a good projection.

Silhouette index

The Silhouette index assigns to each member of a cluster a confidence indicator of membership, known as the *Silhouette width*. The Silhouette width for the i^{th} sample in cluster X_j $(j = 1, 2, \ldots, n)$ is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where $a(i)$ is the average distance between the i^{th} sample and all of the samples included in X_j and $b(i)$ is the minimum average distance between the i^{th} sample

and all of the samples in other clusters[2]. Thus, for a given cluster, X_j, it is possible to calculate a cluster Silhouette S_j, which characterizes the heterogeneity and isolation properties of such a cluster:

$$S_j = \frac{1}{|X_j|} \sum_{i=1}^{|X_j|} s(i)$$

We can use the *Global Silhouette value*, GS, as a measure of the goodness of the projection of a data set with n clusters:

$$GS = \frac{1}{n} \sum_{j=1}^{n} S_j \tag{3.1}$$

In this case the projection with the maximum GS is taken as the optimal projection.

Dunn's Index

This index identifies projections with clusters that are compact and well separated. For a projection of n clusters $X_1, X_2, \ldots, X_n$, Dunn's validation index, D, is defined as:

$$D = \min_{1 \le i \le n} \left\{ \min_{\substack{1 \le j \le n \\ j \ne i}} \left\{ \frac{\delta(X_i, X_j)}{\max\limits_{1 \le k \le n} \{\Delta(X_k)\}} \right\} \right\} \tag{3.2}$$

where $\delta(X_i, X_j)$ defines the distance between clusters X_i and X_j, the inter-cluster distance (the distance measures are discussed below); $\Delta(X_k)$ represents the intra-cluster distance of cluster X_k. This index gives higher values when the inter-cluster distances are high whilst the intra-cluster distances are low. Thus large values of D correspond to good projections.

Davies-Bouldin Index

As with Dunn's index, the Davies-Bouldin index aims at identifying sets of clusters that are compact and well separated; it is defined as:

$$DB = \frac{1}{n} \sum_{i=1}^{n} \max_{i \ne j} \left\{ \frac{\Delta(X_i) + \Delta(X_j)}{\delta(X_i, X_j)} \right\} \tag{3.3}$$

[2]If x is the sample being considered, $b(i) = \min\limits_{k \ne j} \{(1/|X_k|) \sum_{y \in X_k} d(x, y)\}$

where $\delta(X_i, X_j)$, $\Delta(X_i)$, $\Delta(X_j)$ and n are defined as for equation (3.2). Small values of DB correspond to clusters that are compact, and whose centres are far away from each other. Therefore, a good projection gives a low value for the Davies-Bouldin index.

Different methods may be used to calculate inter-cluster and intra-cluster distances [7]. Thirty-six indices based on equations (3.2) and (3.3) were calculated. These indices consist of different combinations of inter-cluster and intra-cluster distance methods. Six inter-cluster distances, $\delta_i,\ 1 \leq i \leq 6$; and 3 intra-cluster distances, $\Delta_i,\ 1 \leq i \leq 3$ were implemented. Thus for example, D_{13}, represents a Dunn's validity index based on an inter-cluster distance, δ_1, and an intra-cluster distance Δ_3. The mathematical definitions of these inter-cluster and intra-cluster distances are described in the following subsection.

Distances used to implement the validation methods

For the calculation of the Dunn's and Davies-Bouldin validity indices, it is possible to use six different inter-cluster distances (see Table 3.2) and three different intra-cluster distances (see Table 3.3). Taking the distance between two samples as the Euclidean distance, the inter-cluster distances that measures the distance between clusters are:

1. **Single linkage distance:** defines the closest distance between two samples belonging to two different clusters.

2. **Complete linkage distance:** represents the distance between the most remote samples belonging to two different clusters.

3. **Average linkage distance:** defines the average distance between all of the samples belonging to two different clusters.

4. **Centroid linkage distance:** reflects the distance between the centres of two clusters.

5. **Average to centroids linkage distance:** represents the distance between the centre of a cluster and all of samples belonging to a different cluster.

6. **Hausdorff metric:** is based on the discovery of a maximal distance from samples of one cluster to the nearest sample of another cluster.

Name	Formula
Single linkage	$\delta_1(S,T) = \min\limits_{\substack{x\in S\\ y\in T}}\{d(x,y)\}$
Complete linkage	$\delta_2(S,T) = \max\limits_{\substack{x\in S\\ y\in T}}\{d(x,y)\}$
Average linkage	$\delta_3(S,T) = \frac{1}{\lvert S\rvert\lvert T\rvert}\sum\limits_{\substack{x\in S\\ y\in T}} d(x,y)$
	where $\lvert.\rvert$ represents the cardinality of a set
Centroid linkage	$\delta_4(S,T) = d(v_s,v_t)$
	where $v_s = \frac{1}{\lvert S\rvert}\sum_{x\in S} x$ and $v_t = \frac{1}{\lvert T\rvert}\sum_{x\in T} x$
Average to centroids linkage	$\delta_3(S,T) = \frac{1}{\lvert S\rvert+\lvert T\rvert}\left(\sum_{x\in S} d(x,v_t) + \sum_{y\in T} d(y,v_s)\right)$
Hausdorff metric	$\delta_6(S,T) = \max\{\delta(S,T),\delta(T,S)\}$
	where $\delta(S,T) = \max_{x\in S}\{\min_{y\in T}\{d(x,y)\}\}$
	and $\delta(T,S) = \max_{y\in T}\{\min_{x\in S}\{d(x,y)\}\}$.

Table 3.2: Intercluster distances

The intra-cluster distances, that measure the compactness of the clusters, are:

1. **Complete diameter distance:** represents the distance between the most remote samples belonging to the same cluster.

2. **Average diameter distance:** defines the average distance between all of the samples belonging to the same cluster.

3. **Centroid diameter distance:** reflects the double average distance between all of the samples and the cluster centre.

3.2.2 Comparison of the EPP neural networks

In Section 3.1 we used an artificial dataset to compare the three EPP algorithms and to show that the combined one has a very fast convergence which gets good final results. Here we compare the three algorithms using a real dataset. The data set used is from a scientific study of various forms of algae some of which have been manually identified. Each sample is recorded as an 18 dimensional vector representing the magnitudes of various pigments. Some algae have been identified

Name	Formula
Complete diameter	$\Delta_1(S) = \max_{x,y \in S}\{d(x,y)\}$
Average diameter	$\Delta_2(S) = \frac{1}{\lvert S\rvert \cdot (\lvert S\rvert -1)} \sum_{\substack{x,y \in S \\ x \neq y}} d(x,y)$
Centroid diameter	$\Delta_3(S) = \frac{2}{\lvert S\rvert} \sum_{x \in S} d(x, v_s)$
	where $v_s = \frac{1}{\lvert S\rvert} \sum_{x \in S} x$.

Table 3.3: Intracluster distances

as belonging to specific classes which are numbered 1 to 9 (72 samples). Others are as yet unclassified and these are labelled 0 (46 samples). In the experiments only the labelled data were used. Figure 3.6 shows a projection of these samples onto the first two Principal Components.

We perform two groups of experiments. The first determines how fast the algorithms converge and the second measures how good the projections are. The experiments consisted of 100 training simulations for each method, each using 80000 iterations. When the training process finished, the different indices were calculated for the obtained projection. In the second group of experiments the training ends normally. But in the first group, the training was aborted after 16000 iterations (that is the annealing process was as though 80000 iterations were to be executed, but the process was stopped after only 20% of the iterations were done). By stopping early, we aim to get a measure of the speed of convergence of the algorithms — a faster algorithm will, *ceteris paribus*, be closer to the optimal than a slower algorithm.

The left parts of Tables 3.4 and 3.5 show the values of the indices. Table 3.4 was obtained by calculating the indices for the projections obtained after aborting the process. In Table 3.5, we show the values when the full training process of 80000 iterations was performed. In both cases, the combined algorithm gives better results than other two. That is the same conclusion we arrived at in Section 3.1, but this time we have been using a real data set.

As described by Bolshakova and Azuaje [15]:

> The application of different intercluster/intracluster distance combinations may produce validation indices of different scale ranges. Hence those indices with higher values may have a stronger effect on the calculation of the average index values. ... the average values of the

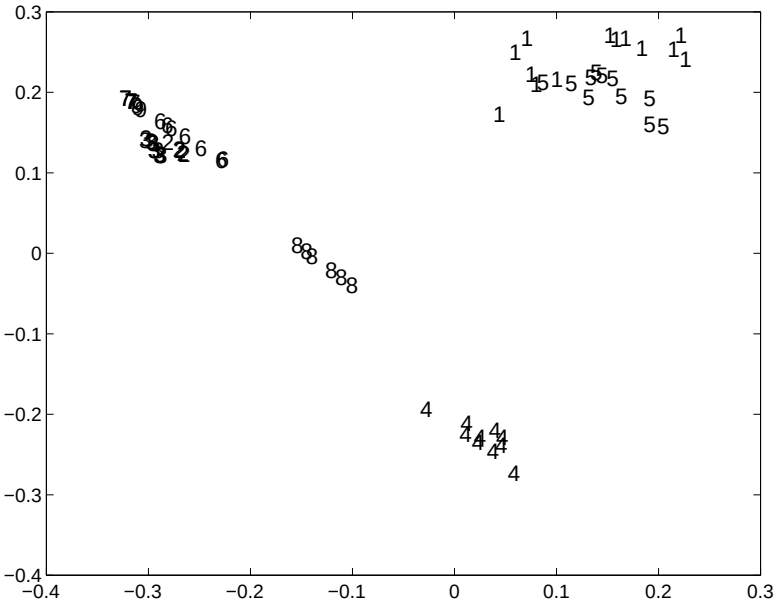


Figure 3.6: Projection of the labelled samples of the algae data set onto the first two principal components.

> Davies-Boulding validation index [*and of the Dunn index as well*] are strongly influenced by the values based on the complete diameter distance.

To solve that problem, they normalize the index values. We have used the same normalisation technique as [15], but applying it to projections rather than partitions. Given the projections obtained from the application of the three algorithms (P_1, P_2, P_3), the normalized Dunn indices, D^*_{ij}, are calculated as:

$$D^*_{ij}(P_A) = \frac{D_{ij}(P_A) - \overline{D}_{ij}}{\sigma_{D_{ij}}}, i = 1, \ldots, 6, j = 1, \ldots, 3 \tag{3.4}$$

$$\overline{D}_{ij} = \frac{1}{3}\sum_{k=1}^{3} D_{ij}(P_k) \tag{3.5}$$

where $D_{ij}(P_A)$ is the value of the Dunn's index for the projection P_A ($A = 1$, Combined algorithm; $A = 2$, Output Function algorithm; $A = 3$, Maximum Likelihood algorithm) when the i^{th} inter-class and j^{th} intra-class distances are used, and $\sigma_{D_{ij}}$ is the standard deviation of $D_{ij}(P_A)$ across the three projections. The normalized values using the previous formula are shown in brackets in the tables.

Another way to measure the goodness of projections is to use an aggregation method based on a weighted voting strategy. The right parts of the tables were obtained from the left parts by replacing the index values by weighted votes, whose values range from 1 to 3. That is, the smallest values for the Davies-Boulding and C indices will give a highest vote to the corresponding projection. The highest values for the Goodman-Kruskal, Silhouette and Dunn indices will give a high vote to the projection with those index values. The combined algorithm gives, on the whole, better results.

Index	Combined	Output	Maximum
C	**0.19 (-1.14)**	0.22 (0.71)	0.22 (0.44)
GK	0.62 (0.52)	**0.63 (0.64)**	0.46 (-1.15)
S	**0.10 (0.74)**	0.09 (0.40)	0.05 (-1.14)
D_{11}	**0.02 (1.10)**	0.01 (-0.25)	0.01 (-0.85)
D_{21}	**0.50 (1.13)**	0.31 (-0.78)	0.35 (-0.35)
D_{31}	**0.24 (1.09)**	0.16 (-0.87)	0.18 (-0.22)
D_{41}	**0.09 (0.80)**	0.08 (0.33)	0.05 (-1.12)
D_{51}	**0.19 (1.13)**	0.14 (-0.75)	0.15 (-0.38)
D_{61}	**0.25 (1.15)**	0.18 (-0.45)	0.17 (-0.70)
D_{12}	**0.03 (1.07)**	0.03 (-0.15)	0.02 (-0.92)
D_{22}	**1.09 (1.12)**	0.73 (-0.80)	0.82 (-0.32)
D_{32}	**0.53 (1.07)**	0.37 (-0.91)	0.43 (-0.16)
D_{42}	**0.19 (0.63)**	0.18 (0.52)	0.11 (-1.15)
D_{52}	**0.42 (1.13)**	0.32 (-0.78)	0.34 (-0.35)
D_{62}	**0.55 (1.14)**	0.42 (-0.39)	0.39 (-0.74)
D_{13}	**0.02 (1.08)**	0.02 (-0.18)	0.01 (-0.90)
D_{23}	**0.78 (1.12)**	0.51 (-0.79)	0.58 (-0.33)
D_{33}	**0.38 (1.08)**	0.26 (-0.90)	0.30 (-0.18)
D_{43}	**0.13 (0.68)**	0.13 (0.47)	0.08 (-1.15)
D_{53}	**0.30 (1.13)**	0.22 (-0.77)	0.24 (-0.36)
D_{63}	**0.40 (1.14)**	0.29 (-0.41)	0.27 (-0.73)
Avg	**0.34 (1.04)**	0.24 (-0.44)	0.25 (-0.61)
DB_{11}	**58.38 (-0.91)**	72.59 (-0.16)	95.93 (1.07)
DB_{21}	1.63 (-0.45)	**1.62 (-0.70)**	1.69 (1.15)
DB_{31}	**3.41 (-0.71)**	3.46 (-0.44)	3.78 (1.14)
DB_{41}	10.08 (-0.30)	**8.35 (-0.82)**	14.77 (1.12)
DB_{51}	4.21 (-0.51)	**4.17 (-0.64)**	4.75 (1.15)
DB_{61}	3.51 (-0.04)	**3.09 (-0.98)**	3.98 (1.02)
DB_{12}	**26.99 (-0.93)**	33.35 (-0.13)	42.86 (1.06)
DB_{22}	0.80 (0.64)	**0.76 (-1.15)**	0.79 (0.52)
DB_{32}	1.62 (-0.50)	**1.61 (-0.65)**	1.70 (1.15)
DB_{42}	4.77 (-0.23)	**3.86 (-0.86)**	6.67 (1.10)
DB_{52}	2.01 (-0.24)	**1.94 (-0.86)**	2.15 (1.10)
DB_{62}	1.65 (0.08)	**1.43 (-1.04)**	1.83 (0.96)
DB_{13}	**37.16 (-0.94)**	46.37 (-0.11)	59.36 (1.05)
DB_{23}	1.09 (0.68)	**1.05 (-1.15)**	1.09 (0.47)
DB_{33}	2.22 (-0.58)	**2.22 (-0.58)**	2.33 (1.15)
DB_{43}	6.58 (-0.24)	**5.35 (-0.86)**	9.22 (1.10)
DB_{53}	2.76 (-0.27)	**2.67 (-0.84)**	2.96 (1.11)
DB_{63}	2.28 (0.08)	**1.98 (-1.04)**	2.53 (0.96)
Avg	**9.51** (-0.30)	10.88 **(-0.72)**	14.35 (1.02)

Ind.	Com.	Out.	Max.
C	3	1	2
GK	2	3	1
S	3	2	1
D_{11}	3	2	1
D_{21}	3	1	2
D_{31}	3	1	2
D_{41}	3	2	1
D_{51}	3	1	2
D_{61}	3	2	1
D_{12}	3	2	1
D_{22}	3	1	2
D_{32}	3	1	2
D_{42}	3	2	1
D_{52}	3	1	2
D_{62}	3	2	1
D_{13}	3	2	1
D_{23}	3	1	2
D_{33}	3	1	2
D_{43}	3	2	1
D_{53}	3	1	2
D_{63}	3	2	1
DB_{11}	3	2	1
DB_{21}	2	3	1
DB_{31}	3	2	1
DB_{41}	2	3	1
DB_{51}	2	3	1
DB_{61}	2	3	1
DB_{12}	3	2	1
DB_{22}	1	3	2
DB_{32}	2	3	1
DB_{42}	2	3	1
DB_{52}	2	3	1
DB_{62}	2	3	1
DB_{13}	3	2	1
DB_{23}	1	3	2
DB_{33}	2	3	1
DB_{43}	2	3	1
DB_{53}	2	3	1
DB_{63}	2	3	1
Avg	2.56	2.13	1.31

Table 3.4: Left: Validity indices for the three EPP neural networks for shortened training. Bold entries represent the optimal value (in brackets the normalized values). Right: Votes for the EPP algorithms.

Index	Combined	Output	Maximum
C	**0.19 (-1.07)**	0.22 (0.91)	0.21 (0.16)
GK	**0.64 (0.62)**	0.63 (0.53)	0.46 (-1.15)
S	**0.14 (1.14)**	0.07 (-0.41)	0.05 (-0.73)
D_{11}	**0.02 (1.14)**	0.01 (-0.43)	0.01 (-0.72)
D_{21}	**0.54 (1.14)**	0.31 (-0.70)	0.35 (-0.44)
D_{31}	**0.26 (1.11)**	0.14 (-0.82)	0.17 (-0.30)
D_{41}	0.06 (0.44)	**0.06 (0.70)**	0.05 (-1.15)
D_{51}	**0.20 (1.13)**	0.12 (-0.77)	0.14 (-0.36)
D_{61}	**0.29 (1.15)**	0.16 (-0.58)	0.16 (-0.58)
D_{12}	**0.04 (1.11)**	0.02 (-0.29)	0.02 (-0.82)
D_{22}	**1.16 (1.15)**	0.76 (-0.64)	0.79 (-0.52)
D_{32}	**0.56 (1.12)**	0.34 (-0.79)	0.40 (-0.34)
D_{42}	0.13 (0.09)	**0.15 (0.95)**	0.11 (-1.04)
D_{52}	**0.43 (1.14)**	0.29 (-0.72)	0.31 (-0.42)
D_{62}	**0.64 (1.15)**	0.39 (-0.51)	0.37 (-0.65)
D_{13}	**0.03 (1.12)**	0.02 (-0.32)	0.01 (-0.80)
D_{23}	**0.84 (1.15)**	0.53 (-0.65)	0.55 (-0.50)
D_{33}	**0.41 (1.13)**	0.24 (-0.79)	0.28 (-0.34)
D_{43}	0.09 (0.22)	**0.10 (0.87)**	0.07 (-1.09)
D_{53}	**0.31 (1.14)**	0.20 (-0.73)	0.22 (-0.41)
D_{63}	**0.46 (1.15)**	0.27 (-0.52)	0.26 (-0.63)
Avg	**0.36 (0.99)**	0.23 (-0.37)	0.24 (-0.62)
DB_{11}	**64.82 (-1.00)**	78.52 (0.01)	92.00 (1.00)
DB_{21}	**1.61 (-0.91)**	1.64 (-0.16)	1.68 (1.07)
DB_{31}	**3.32 (-1.12)**	3.62 (0.30)	3.73 (0.81)
DB_{41}	11.96 (-0.09)	**9.63 (-0.95)**	15.02 (1.04)
DB_{51}	**4.16 (-0.91)**	4.37 (-0.16)	4.70 (1.07)
DB_{61}	**3.33 (-0.61)**	3.35 (-0.55)	3.86 (1.15)
DB_{12}	**29.05 (-1.01)**	35.30 (0.01)	41.27 (0.99)
DB_{22}	0.79 (0.69)	**0.77 (-1.15)**	0.78 (0.45)
DB_{32}	**1.57 (-1.12)**	1.66 (0.33)	1.68 (0.79)
DB_{42}	5.68 (0.04)	**4.42 (-1.02)**	6.80 (0.98)
DB_{52}	**1.99 (-0.64)**	2.00 (-0.51)	2.14 (1.15)
DB_{62}	1.56 (-0.47)	**1.53 (-0.68)**	1.79 (1.15)
DB_{13}	**40.03 (-1.03)**	49.15 (0.06)	56.77 (0.97)
DB_{23}	1.08 (0.97)	**1.06 (-1.03)**	1.07 (0.06)
DB_{33}	**2.15 (-1.13)**	2.27 (0.38)	2.30 (0.75)
DB_{43}	7.80 (0.04)	**6.08 (-1.02)**	9.34 (0.98)
DB_{53}	**2.72 (-0.68)**	2.74 (-0.47)	2.92 (1.15)
DB_{63}	2.15 (-0.48)	**2.11 (-0.67)**	2.46 (1.15)
Avg	**10.32 (-0.53)**	11.68 (-0.40)	13.91 (0.93)

Ind.	Com.	Out.	Max.
C	3	1	2
GK	3	2	1
S	3	2	1
D_{11}	3	2	1
D_{21}	3	1	2
D_{31}	3	1	2
D_{41}	2	3	1
D_{51}	3	1	2
D_{61}	3	1	2
D_{12}	3	2	1
D_{22}	3	1	2
D_{32}	3	1	2
D_{42}	2	3	1
D_{52}	3	1	2
D_{62}	3	2	1
D_{13}	3	2	1
D_{23}	3	1	2
D_{33}	3	1	2
D_{43}	2	3	1
D_{53}	3	1	2
D_{63}	3	2	1
DB_{11}	3	2	1
DB_{21}	3	2	1
DB_{31}	3	2	1
DB_{41}	2	3	1
DB_{51}	3	2	1
DB_{61}	3	2	1
DB_{12}	3	2	1
DB_{22}	1	3	2
DB_{32}	3	2	1
DB_{42}	2	3	1
DB_{52}	3	2	1
DB_{62}	2	3	1
DB_{13}	3	2	1
DB_{23}	1	3	2
DB_{33}	3	2	1
DB_{43}	2	3	1
DB_{53}	3	2	1
DB_{63}	2	3	1
Avg	2.69	1.97	1.33

Table 3.5: Left: Validity indices for the three EPP neural networks (full training). Bold entries represent the optimal value (in brackets the normalized values). Right: Votes for the EPP algorithms.

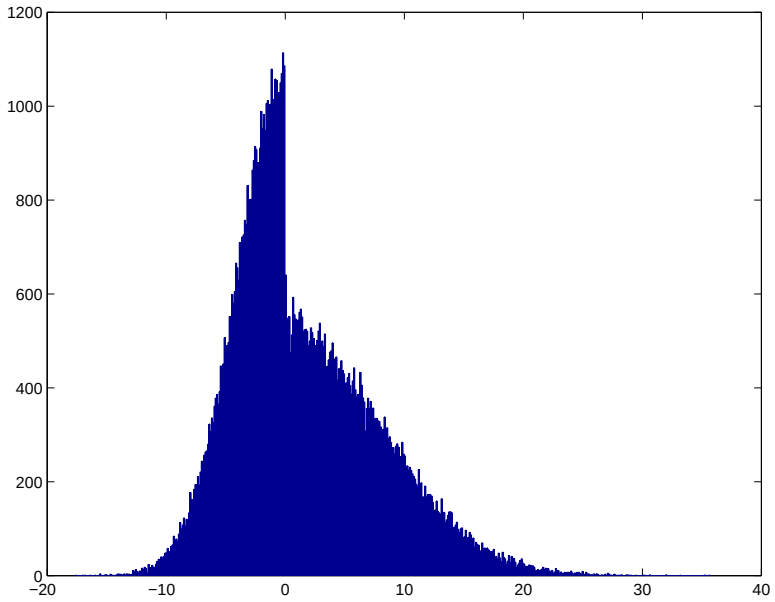


Figure 3.7: Histogram of samples of skewed distribution.

3.3 Identification of Skewness

So far our interest has been defined in terms of kurtosis: outliers are often signalled by positive kurtosis and clusters by negative kurtosis. However this is not the only structure to be found in data sets: we mentioned in Section 2.2.2 that skewness could be found with the Output Functions Algorithm [55] with $f(y) = y^2$. More recently, it has been suggested by MacDonald and Fyfe [119] that we can use $f(y) = -\cos(y)$ (whose Taylor expansions $-1+(1/2)y^2-(1/24)y^4+(1/720)y^6\ldots$) in this algorithm to search for skewness. We first compare these methods on an artificial data set exhibiting skewness; a histogram of samples is shown in Figure 3.7.

The two output functions are compared in Figure 3.8. The first interesting conclusion we can draw from this study is that the $-\cos()$ function, while perhaps more stable than simple squaring does not converge so quickly or reliably.

Until now, skewness has not been identified by the Maximum Likelihood algorithm since all learning rules used to date have been based on the symmetric family of distributions described above. However, if we consider a skewed (centred) distribution: it will typically have a large probability mass close to zero for positive (negative) values and much longer tails but no great lump of probability

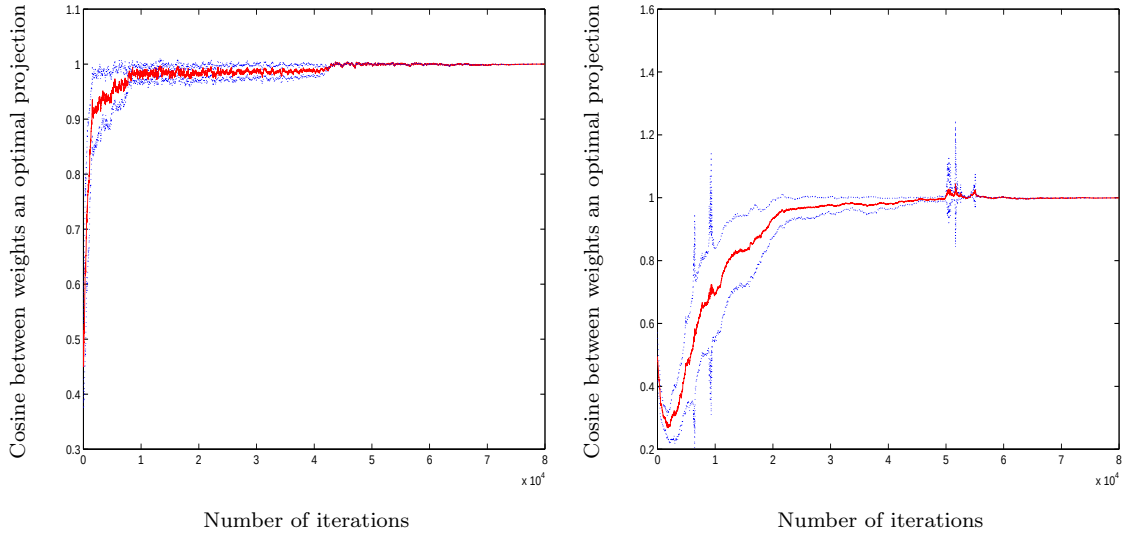


Figure 3.8: The *left* diagram shows convergence of 81 out of 100 simulations on skewed data using y^2 as the output function. 18 simulations were unstable and 1 did not converge after 80000 iterations. The *right* diagram shows convergence of 94 simulations which used the $-\cos()$ function; 1 was unstable and 5 more did not converge after 80000 iterations.

mass for negative (positive) values. Therefore we suggest an algorithm so that

$$\Delta w_{ij} = \begin{cases} \eta y_i \operatorname{sign}(e_j) & \text{if } e_j < 0 \\ \eta y_i \operatorname{sign}(e_j)|e_j|^2 & \text{if } e_j > 0 \end{cases} \tag{3.6}$$

Thus we are modelling a skewed distribution with values of $p = 1$ on one side of the origin and $p = 3$ on the other. Of course, the skewness may be the other way round in which case we reverse the equations. Results are shown in Figure 3.9. We see that this algorithm is only partly successful: although we achieve very fast convergence in all cases, we do not achieve more than 80% accuracy on this skewed data set.

As before, we attempted to combine the two algorithms, however we were unable with the same learning rate to get convergence on this data set with a combined algorithm.

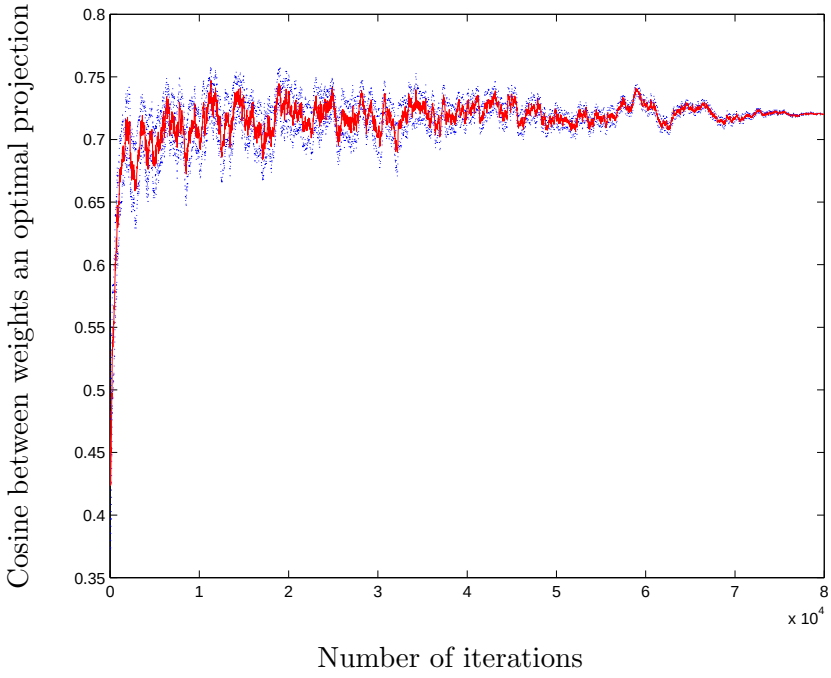


Figure 3.9: Convergence of the algorithm using the new skew Maximum Likelihood method in 100 simulations.

3.4 Initializing EPP Networks with Andrews' Curves

In this section we investigate a method of biasing the learning in an EPP network so that the network is most likely to find directions which maximize the selected statistic. Essentially we do this by initializing the weights of the EPP network to specific values which make it most likely that the network will reach the global optimum rather than any local optima.

To study the improvements that could be obtained using the Andrews' curves we performed many experiments using artificial data sets as in Section 3.1. In four experiments we used 10 dimensional inputs, 9 independent Gaussian dimensions and one platykurtotic dimension obtained from a Gaussian distribution which was then randomly changed by adding or subtracting the same quantity to the samples. This gives a bimodal distribution with two clusters. Following that procedure, 300 points were obtained and then the data was sphered. In the fifth experiment we used 19 Gaussians instead of 9, obtaining in this way a more complex data set. Each experiment comprised 20 simulations in which the networks

were trained using 20000 samples and initialized using the weights obtained from the Andrews' curves, and 20 more simulations with random initial weights (between zero and one). To present the results the mean of the 20 simulations is used. In all cases but one (see below) the value of the initial learning rate was 0.01 which was annealed to zero during the course of the simulation. These artificial data sets are convenient since we know what direction we are searching for (e.g. (1,0,0,0, ...,0)) and it is very easy to check convergence.

As we have mentioned, the EPP networks that we use get the most accurate results when sphering is applied to the data. On the contrary, the Andrews' curves obtained using the sphered data lack the necessary structure that make them useful as a visualization tool. That does not happen when the raw data is used. So in the experiments, that we present below, the Andrews' curve over the original data are used. Then the vector obtained after selecting an interesting t value is transformed in accordance with the process of sphering before being used to initialize the network weights. The network operation is $y = \mathbf{wz} = \mathbf{w}S\mathbf{x}$, since it is going to be fed with sphered data. While the Andrews's curves gives for each value t the value $y = \mathbf{f}_1(t)\mathbf{x}$, where $\mathbf{f}_1(t)$ is the vector $\left(1/\sqrt{2}, \cos(t), \sin(t), \cos(2t), \sin(2t), \ldots\right)$. So the transformation that gives the initial weights is $\mathbf{w} = \mathbf{f}_1(t)S^{-1}$. The t value is obtained by visually selecting a projection in the Andrews curves of the data which makes the identification of clusters clearer.

Thus our conjecture is that we can identify some structure in the Andrews' plots and use this as an initial condition for an EPP network.

In the first experiment, we used the Output Functions EPP algorithm [55] with the function $f(y) = \tanh(y)$ in the learning rule. Figure 3.10 shows the good convergence obtained when the Andrews' curves are used to initialize the network weights. With random weights the convergence is very poor i.e. often non-existent.

Now the tanh() function was used as a stable approximation to $-y^3$. We may now report that with a second set of experiments using $-y^3$ and a learning rate of 0.0001, we achieved 100% convergence when using the Andrews' Curves initialization (Figure 3.11) and 100% NaN errors when not.

The aim of the third experiment was to investigate the effect of this initialization on the Maximum Likelihood EPP algorithm. We can see in Figures 3.12 and 3.13 that with the initialization of the weights, little is gained. Indeed, this EPP algorithm actually moves the weights away from the optimal

values so that what was taken, in previous experiments, to be lack of accuracy on the part of the algorithm is actually due to the algorithm learning something other than that which it was designed to learn. We see that whether we initialize with the Andrews' curves values or randomly, the weights converge to approximately the same values.

The fourth experiment makes use of the learning rule which is a combination of the Output Function and the Maximum Likelihood learning rules. As we expected the influence of the initialization negligible, since this learning rule exhibits a very good convergence even with random initial weights.

The last experiment uses a more complex data set than the previous ones. On this occasion we tried to identify a bimodal distribution immersed in 19 Gaussian distributions. Again the advantages of the initialization can be appreciated. Figure 3.14 shows the results with the initialization from the Andrews' Curves, in Figure 3.15 the 8 best results without the initialization are shown.

3.5 Conclusions

We have, in this chapter, reviewed two neural implementations of Exploratory Projection Pursuit and shown that both are capable of performing this difficult statistical task. We showed empirically that the Output Functions EPP algorithm is slower to converge but gives us more accurate results. However, convergence in the Output Functions algorithm tends to cease when the weights become orthonormal (the constraint condition is satisfied). If this happens before the most interesting direction has been found, and this is often the case, we will have a less than perfect projection. Thus anything which can be used to speed up convergence is useful.

The Maximum Likelihood EPP algorithm on the other hand does exhibit very fast convergence but with much less accuracy. Therefore anything which helps improve the accuracy of this algorithm is to be welcomed. This seems to be a marriage made in heaven since each side of the combined algorithm has strengths which the other needs and conversely each side has weaknesses which the other is in a position to overcome.

It is also worth noting that the combined algorithm cannot be derived easily from a minimization of error criterion since the feedback is from a linear neuron and the nonlinearity in the outputs appears directly in the learning rule. Therefore the combined algorithm remains, at present, merely an heuristic which works

well in practice. Besides, the combined algorithm converges faster and gives better two-dimensional projections than the other two even when a real data set is used, and objective comparison criteria applied.

We can highlight also that the process followed in the comparison using the real data set can be applied to the comparison of other types of EPP algorithms, avoiding the bias of using only few projections, and the subjective judgement of the person who compares the goodness of the projections.

We have also extended the Maximum Likelihood method so that it can now identify skewed distributions though this met with only limited success: we have very fast convergence to approximately the correct direction but the convergence though stable is very approximate.

We have, finally, investigated the improvements that can be achieved using initial weights other than random weights. The method that we used to do the initialization is based on Andrews' curves that enables the user to choose a promising projection where the EPP network can start its search.

We have shown how, with appropriate initialization of the weights, an algorithm with very bad convergence can achieve a very good convergence. The initialization even enables us to use algorithms that, without it, do not exhibit convergence. Interestingly, the results of Figures 3.12 and 3.13 suggest that the results which we had previously ascribed to poor convergence actually are due to the algorithm responding to statistics other than that which it is designed for. This aspect has thrown new light on the convergence of these EPP algorithms and is worthy of future theoretical investigation.

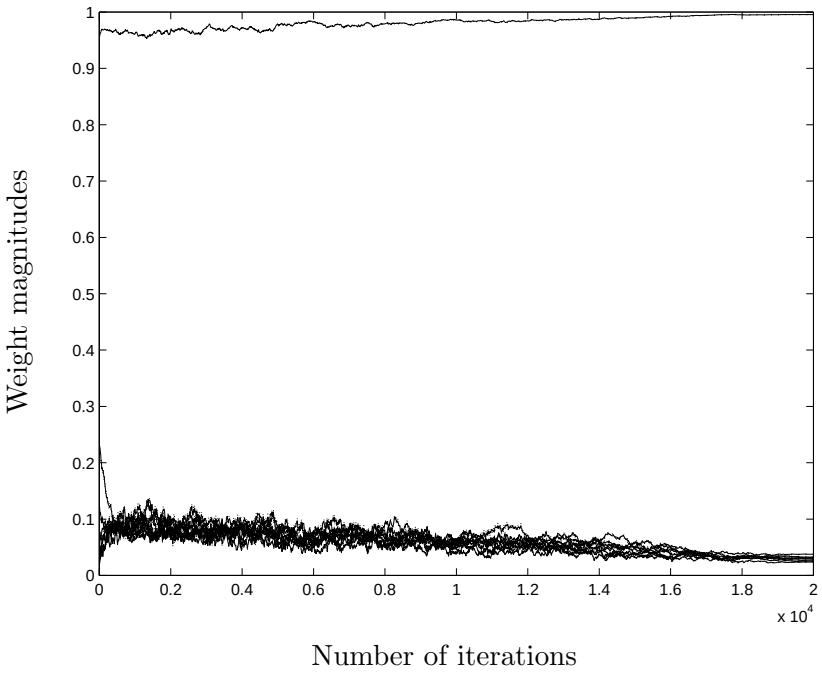


Figure 3.10: The mean of 20 simulations that use the Output Function EPP Algorithm ($f(y) = \tanh(y)$) using weight initialization from specific Andrews' Curves values (magnitude of 10 weights—not only the cosine as previously). We also show one standard deviation either side of the mean (here and in Figures 3.11 to 3.14 the standard deviation is almost no visible because its low value).

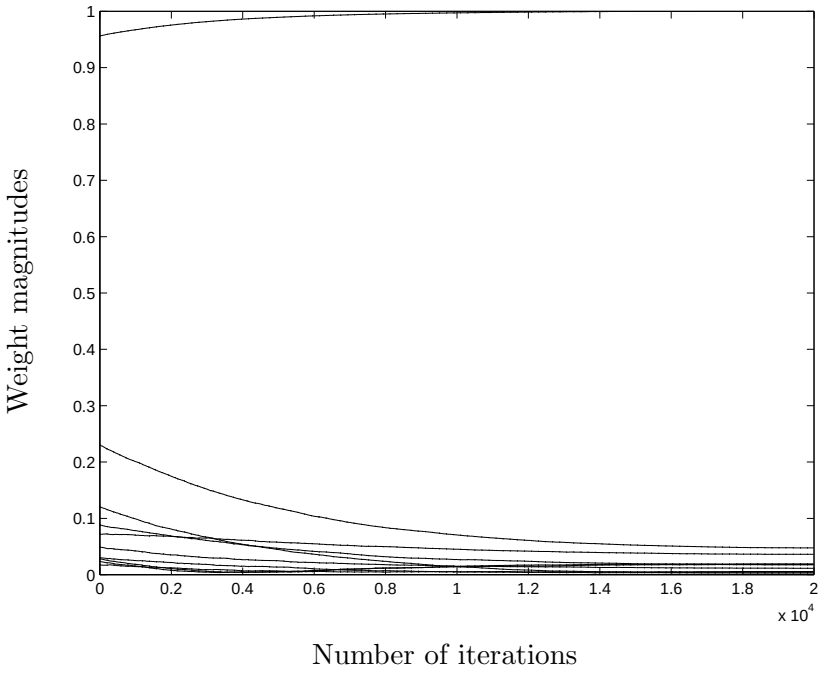


Figure 3.11: The mean of 20 simulations that use the Output Function EPP Algorithm ($f(y) = -y^3$) using weight initialization from specific Andrews' Curves values (magnitude of 10 weights). We also show one standard deviation either side of the mean.

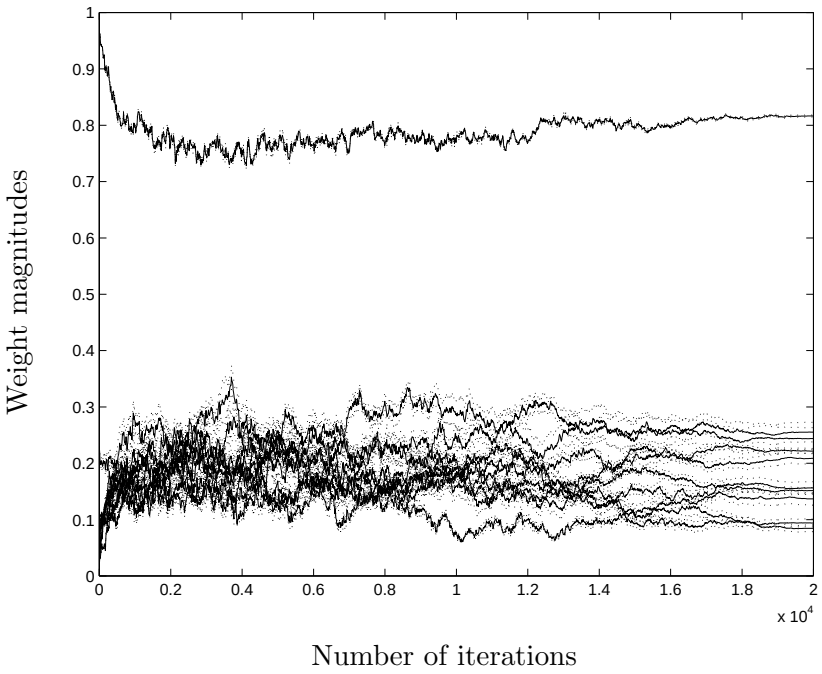


Figure 3.12: The convergence of the 20 simulations that use the Maximum Likelihood EPP Algorithm using the weight initialization from the Andrews' Curves (magnitude of 10 weights).

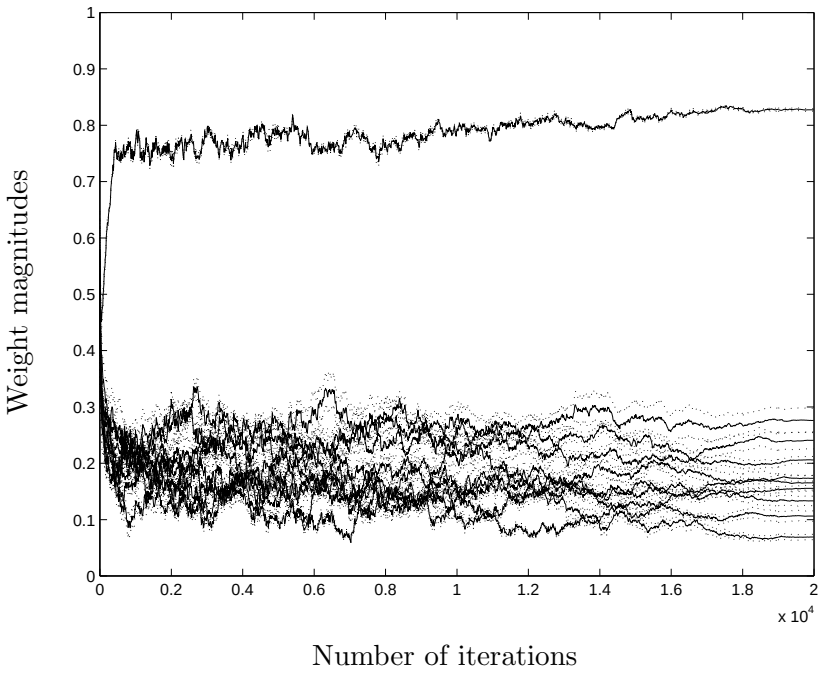


Figure 3.13: The convergence of the 20 simulations that use the Maximum Likelihood EPP Algorithm initialising with random weights (magnitude of 10 weights).

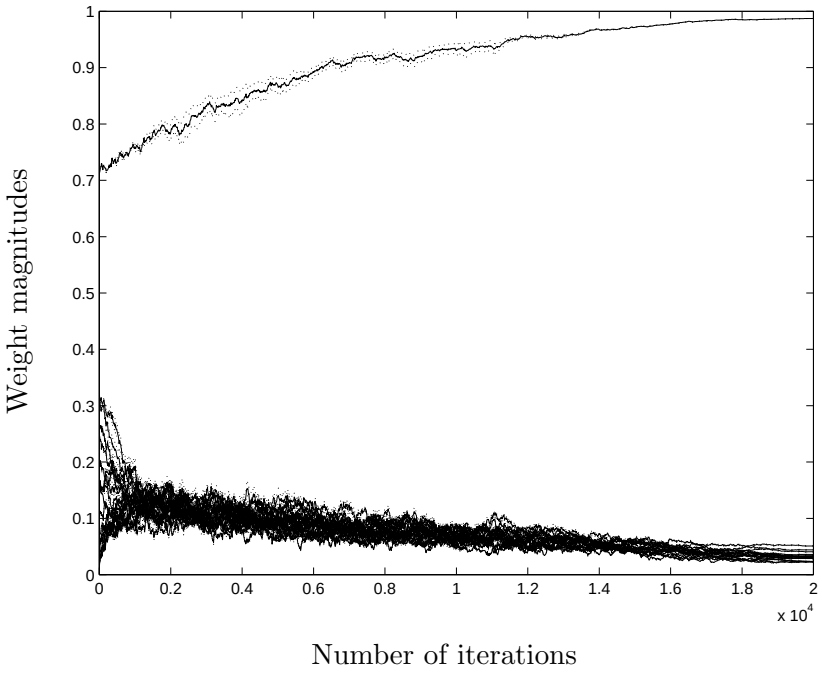


Figure 3.14: Convergence of the Output Function algorithm with $f(y) = \tanh(y)$ using 19 Gaussian distributions and one bimodal distributions using the Andrews weight initialization. We also show one standard deviation either side of the mean (magnitude of 10 weights).

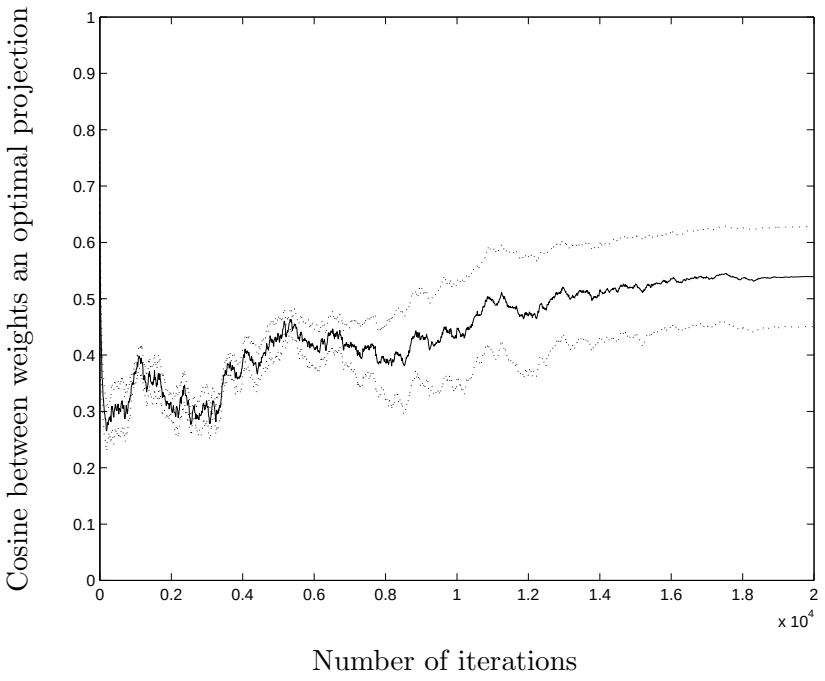


Figure 3.15: Convergence of the Output Function algorithm with $f(y) = \tanh(y)$ using 19 Gaussian distributions and one bimodal distributions (cosine of the angle between the network weights and the optimal weight). The mean of the best 8 experiments (and one standard deviation either side) when the weights are randomly initialized.

Chapter 4

Curve Based Exploratory Data Analysis

A strong desire of all data analysts is to have the ability to visualize data. Often this means taking a low dimensional projection of a data set and looking in turn at a variety of one, two or, at most, three dimensional projections of the data. The previous chapter discussed the automatic search for those projections. An alternative is to transform the data in some way in order to make the data's properties in some way visible via the transformation. Therefore the transformation has to maintain some inherent properties of the data if we are to be able to identify some inherent characteristics of the data after transformation. In this chapter we present a variation of Andrews' curves which performs that kind of transformation.

We start with a review of Wegman's curves, an extension to Andrews' curves that allow us to construct a two dimensional grand tour. We give a new characterization of Wegman's curves, and by reformulating Wegman's contribution, we are able to extend that visualization method to three dimensions which makes the identification of structure very much easier. Then, we analyze the properties of this new transformation and compare it with Wegman's curves. Finally we illustrate how to use this new method in analyzing a real data set, to identify clusters and outliers. Indeed the use of this method enables us to discover new information about the structure in a data set.

4.1 A New Perspective on Wegman's Algorithm

In Section 2.4.2 we presented the Andrews' Curves, obtained through the projection of multidimensional points onto the vector

$$\left(\frac{1}{\sqrt{2}}, \sin(t), \cos(t), \sin(2t), \cos(2t), \ldots\right)$$

Wegman and Solka [180] discuss the benefits (mainly computational) of using a slightly different projection from Andrews, namely that onto

$$\begin{aligned}
\mathbf{w}_1 &= \sqrt{\frac{2}{d}}\Big(\sin(\lambda_1 t), \cos(\lambda_1 t), ..., \sin(\lambda_{\frac{d}{2}} t), \cos(\lambda_{\frac{d}{2}} t)\Big) \\
\mathbf{w}_2 &= \sqrt{\frac{2}{d}}\Big(\cos(\lambda_1 t), -\sin(\lambda_1 t), ..., \cos(\lambda_{\frac{d}{2}} t), -\sin(\lambda_{\frac{d}{2}} t)\Big)
\end{aligned}$$

with the λ_j linearly independent over the rationals. Clearly $(\mathbf{w}_1, \mathbf{w}_2)$ form a set of 2 orthonormal basis vectors. If we define

$$\begin{aligned}
y_1 = \mathbf{w}_1^T \mathbf{x} &\propto x_1 \sin(\lambda_1 t) + x_2 \cos(\lambda_1 t) + ... + x_d \cos(\lambda_{\frac{d}{2}} t) \\
y_2 = \mathbf{w}_2^T \mathbf{x} &\propto x_1 \cos(\lambda_1 t) - x_2 \sin(\lambda_1 t) + ... - x_d \sin(\lambda_{\frac{d}{2}} t)
\end{aligned}$$

then we have a two dimensional display on which to project $\mathbf{x}$ so that we can look for structure by eye[1]. Visually from this projection, we can identify clusters of points which are nearby and whose trajectories as we change t (i.e. as we move along the Andrews' Curves) keep close together. When we use these curves in this way we obtain a two dimensional "grand tour" of the data (see Section 2.5 and Asimov [4]). Figure 4.1 shows the obtained curves and a snapshot of the grand tour.

Other points may approach a particular cluster for a brief period of time but will not remain within the cluster throughout the tour. This may be seen as

$$\begin{aligned}
\frac{\partial y_1}{\partial t} &\propto x_1 \lambda_1 \cos(\lambda_1 t) - x_2 \lambda_1 \sin(\lambda_1 t) + ... - x_d \lambda_{\frac{d}{2}} \sin(\lambda_{\frac{d}{2}} t) \\
\frac{\partial y_2}{\partial t} &\propto -x_1 \lambda_1 \sin(\lambda_1 t) - x_2 \lambda_1 \cos(\lambda_1 t) - ... - x_d \lambda_{\frac{d}{2}} \cos(\lambda_{\frac{d}{2}} t)
\end{aligned}$$

with similar patterns holding at higher orders of derivatives. The points within

[1] If the data set has an odd number of dimensions, we simply add an extra dimension with value 0 to get an even number of dimensions.

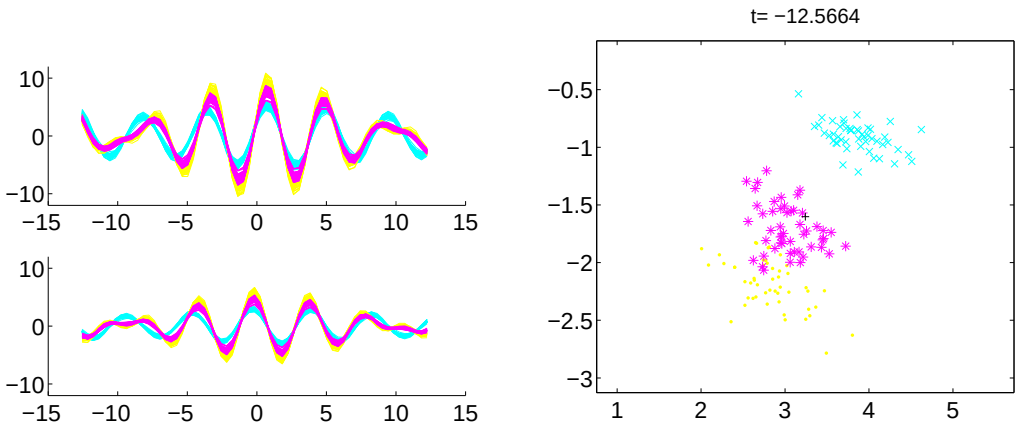


Figure 4.1: *Left*: the two Wegman's curves for the iris data set ($-4\pi \leq t \leq 4\pi$). *Right*: a snapshot of the grand tour.

the cluster have a characteristic dance associated with the joint behaviour of the rates of change which are determined by the derivatives which are sinusoids. Now since we can identify clusters from the position and motion of individual points, this suggests a second projection might be useful and so we now investigate a different set of 2 orthogonal basis vectors. We define

$$\begin{aligned} y_1 = \mathbf{w}_1^T\mathbf{x} &\propto x_1\sin(\lambda_1 t) + x_2\cos(\lambda_1 t) + \ldots + x_d\cos(\lambda_{\frac{d}{2}} t) \\ y_2 = \frac{\partial \mathbf{w}_1^T}{\partial t}\mathbf{x} &\propto x_1\lambda_1\cos(\lambda_1 t) - x_2\lambda_1\sin(\lambda_1 t) + \ldots - x_d\lambda_{\frac{d}{2}}\sin(\lambda_{\frac{d}{2}} t) \end{aligned}$$

where λ_i are now integers as they were with the Andrews original curves. We will call this basis the *derivative basis* in the following. Since

$$\left|\frac{\partial \mathbf{w}_1^T}{\partial t}\right|^2 = \lambda_1^2 + \lambda_2^2 + \ldots + \lambda_{\frac{d}{2}}^2 \tag{4.1}$$

we may readily re-normalize these vectors to get a set of orthonormal vectors the second of which differs slightly from Wegman's basis in that each term is related by

$$(\mathbf{w}_2)_i(\text{derivative}) = \sqrt{\frac{d}{2}}\frac{\lambda_i}{\sqrt{\sum_{i=1}^{\frac{d}{2}}\lambda_i^2}}(\mathbf{w}_2)_i(\text{ Wegman }) \tag{4.2}$$

where we have used $(\mathbf{w}_2)_i$ to denote the i^{th} element of the vector $\mathbf{w}_2$. Therefore the Wegman's basis is a special case of the derivative basis discussed herein. In practice, subjectively we have seen little difference between the projections of the

data onto the two bases.

The derivative basis also gives us an insight into the characteristic dance of points in a cluster: such points appear to move about in a group as though joined together by a set of springs. Let $\bar{\mathbf{x}} = \{\bar{x}_1, \bar{x}_2, ..., \bar{x}_d\}$ be the mean vector of a cluster of data points which have some underlying relation e.g. they all belong to a specific class. Let $\bar{\mathbf{x}}$ be mapped to $\overline{y_1}$ using Andrews' Curves so that

$$\bar{y}_1 = \bar{x}_1 \sin(\lambda_1 t) + \bar{x}_2 \cos(\lambda_1 t) + ...\bar{x}_d \cos(\lambda_{\frac{d}{2}} t) \tag{4.3}$$

Now consider a specific data point, $\mathbf{x}^1$, a member of this cluster, which is mapped to y_1^1. Let

$$\mathbf{x}^1 = \bar{\mathbf{x}} + \epsilon^1 = \{\bar{x}_1, \bar{x}_2, ..., \bar{x}_d\} + \{\epsilon_1^1, \epsilon_2^1, ..., \epsilon_d^1\} \tag{4.4}$$

Then

$$y_1^1 = \bar{y}_1 + \epsilon_1^1 \sin(\lambda_1 t) + \epsilon_2^1 \cos(\lambda_1 t) + ...\epsilon_d^1 \cos(\lambda_{\frac{d}{2}} t) \tag{4.5}$$

Now the distance of y_1^1 from $\bar{y}_1$ is dominated by those ϵ_i^1 terms which have corresponding trigonometric terms tending to 1 i.e. where $|\cos(\lambda_i t)| \to 1$ i.e. $\lambda_i t \to 0, \pi, 2\pi$ etc or $|\sin(\lambda_i t)| \to 1$ i.e. $\lambda_i t \to \pi/2, 3\pi/2$ etc. But these terms are exactly the terms where $\frac{\partial w}{\partial t} \to 0$ i.e. there is liable to be a low rate of change of their position. Thus we tend to see groups moving in a relatively fixed position for reasonably long spells as we change t; it is not too fanciful to describe the resulting movement as a gentle dance.

Perhaps some impression of the dance might be found in Figure 4.2 where we have plotted two classes from an astronomical data set composed of the spectral signatures of a number of asteroids. We see that both classes occupy the same space but the shapes of their trajectories are very different: class 2 (left diagram) is much busier than class 8 (right diagram) because the data points are changing position much more quickly for the given values of t (we show $-\pi < t < \pi$ in the figure).

4.2 Extending the Derivative Curves

Now the Andrews' Curves were derived during the infancy of computational power. We now have very much more powerful machines and, in particular, visual representation on screen is much more sophisticated than it was in those times. Specifically, we now have hardware accelerators which will allow real time

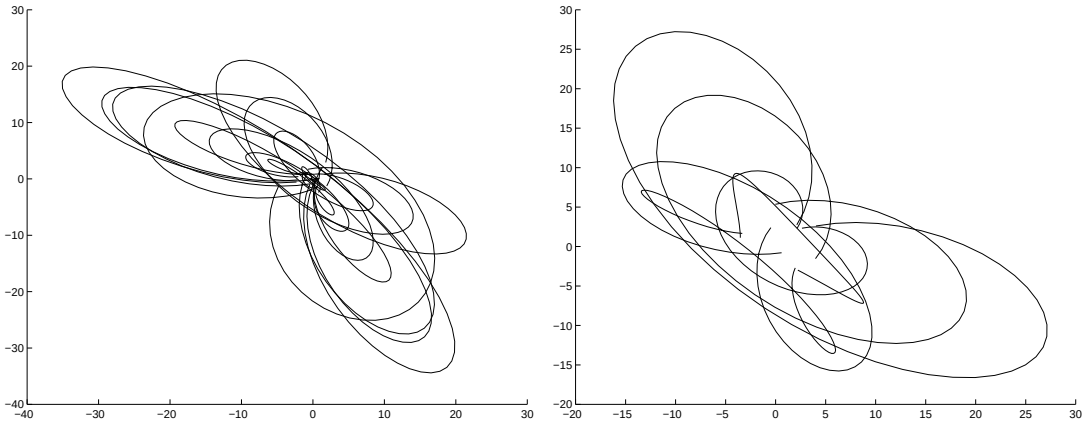


Figure 4.2: Left: Trajectories of members of class 2 (astronomical data set) for $-\pi < t < \pi$. Right: Trajectories of class 8 for $-\pi < t < \pi$.

plotting of three dimensional projections of literally millions of points in real time. Given that the human visual system tends to operate in a three dimensional universe, and has no difficulty in determining the third dimension even when only two are really available (as on a computer monitor), it seems natural to extend the derivative curves to three dimensional representations. In fact, our subjective findings are that this facilitates the extraction of structure from high dimensional data sets by human observers.

Of course this perception of the projections of data points moving in the plane can be extended to data points moving in space so that now $y_i = f(t, s)$ such as

$$y_1 = \mathbf{w}_1^T \mathbf{x} \propto x_1 \cos(\lambda_1 t) \cos(\mu_1 s) + x_2 \cos(\lambda_1 t) \sin(\mu_1 s) + x_3 \sin(\lambda_1 t) + ... \quad (4.6)$$

$$y_2 = \mathbf{w}_2^T \mathbf{x} \propto x_1 \sin(\lambda_1 t) \cos(\mu_1 s) + x_2 \sin(\lambda_1 t) \sin(\mu_1 s) - x_3 \cos(\lambda_1 t) ... \quad (4.7)$$

$$y_3 = \mathbf{w}_3^T \mathbf{x} \propto x_1 \sin(\mu_1 s) - x_2 \cos(\mu_1 s) + x_3 * 0 + ... \quad (4.8)$$

where we have the implicit requirement that the number of terms in each expansion is a multiple of 3 rather than 2 as previously. Note that the second curve is the derivative of the first one with respect to t, and the third is constructed to be orthogonal to the other two and is proportional to the derivative with respect to s. We have omitted the λ_i, μ_i factors for ease of exposition and because in practice there seems to be very little loss of comprehension when we view the movement of the projections without these factors compared to the corresponding movement when we include these factors.

Note that these equations really give three different groups of surfaces in

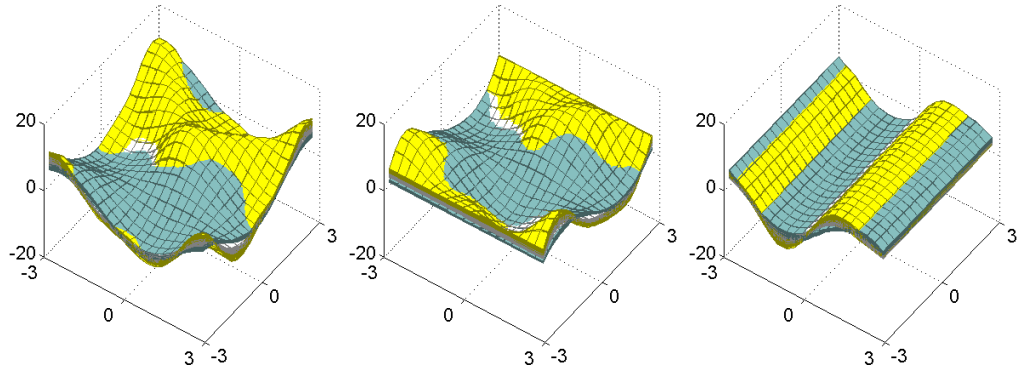


Figure 4.3: The three groups of surfaces for the Iris data set.

3D space but this gives a diagrammatic representation which is very difficult to understand (see Figure 4.3). Thus we prefer to change t and s independently and view the movement of the groups of points through 3D space. We call the curves obtained when the value of t is fixed, *S-slices*, each corresponding to a particular slice of the surface with a specific t value. Similarly, we call *T-slices* the curves obtained when we fix the value of s. Figure 4.4 illustrates this for one group of surfaces. An alternative is to let $t = s$ and view the equations as a curve moving in 3D space (Figure 4.5). One interpretation of this is that the first component represents the (unit) tangent vector to the curve, the second the (unit) normal vector and the third the binormal vector so that these three vectors represent a natural local basis for that space.

4.3 Some properties of the new transformation

As we have discussed in Section 2.4.2, Andrews' curves have the property of distance preservation. That is, the distance between two curves is proportional to the Euclidean distance between the points used to obtain them. Here we show that modifying slightly the formulation of our surfaces, the first two groups of surfaces possess also the property of distance preservation. This time the distance between two surfaces is proportional to the Euclidean distance between the high dimensional points used to calculate them.

We will demonstrate this for the first kind of surfaces (4.6). First we need to slightly change the definition of the curves, by introducing the coefficient $1/\sqrt{2}$

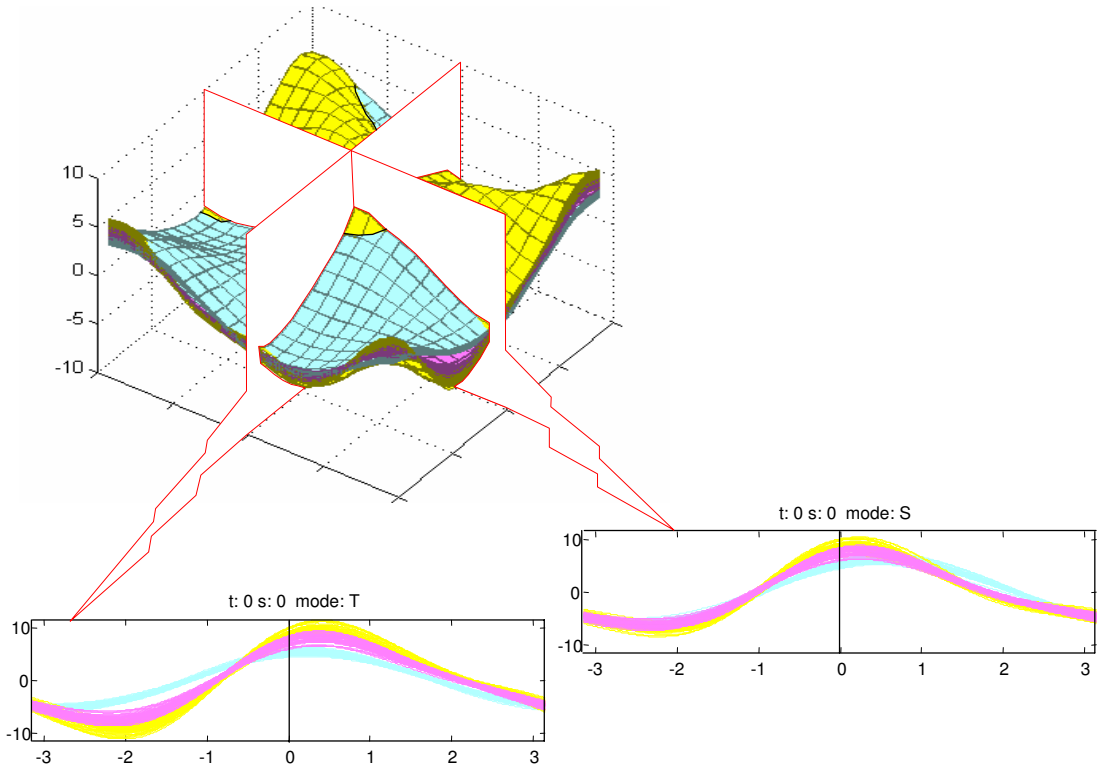


Figure 4.4: One of the surfaces and its T and S slices.

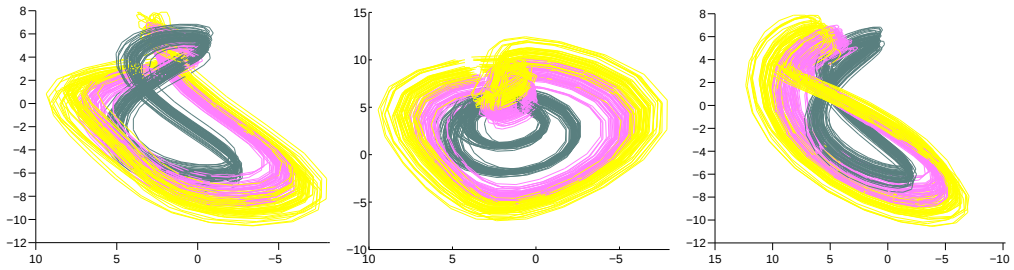


Figure 4.5: Three different perspectives (front, top and side views) of the curves obtained from the Iris data set when $t = s$.

in all coordinates which are multiple of three, so we have:

$$\mathbf{w}_1 = \Big(\cos(\lambda_1 t)\cos(\mu_1 s), \cos(\lambda_1 t)\sin(\mu_1 s), \big(1/\sqrt{2}\big)\sin(\mu_1 s), \cos(\lambda_2 t)\cos(\mu_2 s), \ldots\Big)$$

We define the distance between two surfaces as:

$$V = \iint_S \left[f_{\mathbf{x}}(t,s) - f_{\mathbf{y}}(t,s)\right]^2 dt\, ds$$

where $S = [-\pi, \pi] \times [-\pi, \pi]$.

First, taking into account the definition of $f(t,s)$ we have:

$$V = \iint_S \Big((x_1 - y_1)\cos(\lambda_1 t)\cos(\mu_1 s) + (x_2 - y_2)\cos(\lambda_1 t)\sin(\mu_1 s) + (x_3 - y_3)\frac{1}{\sqrt{2}}\sin(\lambda_1 t) + \ldots\Big)^2 dt\, ds$$

We use the definition of S to get

$$\begin{aligned} V = \int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\Big((x_1 - y_1)^2\cos^2(\lambda_1 t)\cos^2(\mu_1 s) + (x_1 - y_1)(x_2 - y_2)\cos^2(\lambda_1 t)\cos(\mu_1 s)\sin(\mu_1 s) + (x_1 - y_1)(x_3 - y_3)\cos(\lambda_1 t)\cos(\mu_1 s)\frac{1}{\sqrt{2}}\sin(\lambda_1 t) + \ldots \\ (x_2 - y_2)(x_1 - y_1)\cos^2(\lambda_1 t)\sin(\mu_1 s)\cos(\mu_1 s) + (x_2 - y_2)^2\cos^2(\lambda_1 t)\sin^2(\mu_1 s) + (x_2 - y_2)(x_3 - y_3)\cos(\lambda_1 t)\sin(\mu_1 s)\big(1/\sqrt{2}\big)\sin(\lambda_1 t) + \ldots \\ (x_3 - y_3)(x_1 - y_1)\big(1/\sqrt{2}\big)\sin(\lambda_1 t)\cos(\lambda_1 t)\cos(\mu_1 s) + (x_3 - y_3)(x_2 - y_2)\big(1/\sqrt{2}\big)\sin(\lambda_1 t)\cos(\lambda_1 t)\sin(\mu_1 s) + \\ (x_3 - y_3)^2(1/2)\sin^2(\lambda_1 t) + \ldots\Big)dt\, ds \end{aligned}$$

Now, using the orthogonal relations for the sin and cos (see Table 4.1), we obtain

$$\int_{-\pi}^{\pi} \sin^2(nt)dt = \int_{-\pi}^{\pi} \cos^2(nt)dt = \pi \quad \forall n \in \mathbb{N}$$

$$\int_{-\pi}^{\pi} \sin(nt)dt = \int_{-\pi}^{\pi} \cos(nt)dt = 0 \quad \forall n \in \mathbb{N}$$

$$\int_{-\pi}^{\pi} \sin(nt)\sin(mt)dt = \int_{-\pi}^{\pi} \cos(nt)\cos(mt)dt = 0 \quad \forall n, m \in \mathbb{N} \wedge n^2 \neq m^2$$

$$\int_{-\pi}^{\pi} \sin(nt)\cos(mt)dt = \int_{-\pi}^{\pi} \cos(nt)\sin(mt)dt = 0 \quad \forall n, m \in \mathbb{N} \wedge n^2 \neq m^2$$

Table 4.1: Orthogonal relations for the sin and cos.

$$\begin{aligned} V = \int_{-\pi}^{\pi} \Big(& (x_1 - y_1)^2 \pi \cos^2(\mu_1 s) + (x_1 - y_1)(x_2 - y_2)\pi \cos(\mu_1 s)\sin(\mu_1 s) + \dots \\ & \dots + (x_2 - y_2)(x_1 - y_1)\pi \sin(\mu_1 s)\cos(\mu_1 s) + (x_2 - y_2)^2 \pi \sin^2(\mu_1 s) + \dots \\ & \dots + (x_3 - y_3)^2 (1/2)\pi + \dots \qquad\qquad \Big) ds \quad = \\ & (x_1 - y_1)^2\pi^2 + (x_2 - y_2)^2\pi^2 + (x_3 - y_3)^2\pi^2 + \dots \end{aligned}$$

Summing up, we have shown that

$$V = \iint_S [f_{\mathbf{x}}(t,s) - f_{\mathbf{y}}(t,s)]^2 dt\, ds = \pi^2 \sum_i (x_i - y_i)^2 = \pi^2 ||\mathbf{x} - \mathbf{y}||^2$$

With the same change in the second kind of surfaces (4.7) we obtain a similar relation.

As Figure 4.6 graphically shows, other properties that the surfaces hold are (this time even for the third surface):

- The representation yields a one-dimensional projection when we simultaneously fix the values of s and t.

- The surface representation preserves means. If $\bar{\mathbf{x}}$ is the mean of a set of n multivariate observations $\mathbf{x}_i$, the surface corresponding to $\bar{\mathbf{x}}$ is the pointwise

mean of the surfaces corresponding to the n observations:

$$f_{\bar{\mathbf{x}}}(t,s) = \frac{1}{d}\sum_{i=1}^{d} f_{\mathbf{x}_i}(t,s)$$

By definition of $f_{\bar{\mathbf{x}}}(t,s)$

$$f_{\bar{\mathbf{x}}}(t,s) = \bar{x}_1 \cos(\lambda_1 t)\cos(\mu_1 s) + \bar{x}_2 \cos(\lambda_1 t)\sin(\mu_1 s) + \bar{x}_3 \sin(\lambda_1 t) + \ldots$$

By definition of the mean

$$f_{\bar{\mathbf{x}}}(t,s) = \left(\frac{1}{d}\sum_{i=1}^{d} x_{i1}\right)\cos(\lambda_1 t)\cos(\mu_1 s) + \left(\frac{1}{d}\sum_{i=1}^{d} x_{i2}\right)\cos(\lambda_1 t)\sin(\mu_1 s) + \left(\frac{1}{d}\sum_{i=1}^{d} x_{i3}\right)\sin(\lambda_1 t) + \ldots$$

where x_{ij} is the j^{th} coordinate of the i^{th} observation. Now using the associative and distributive properties we have

$$\begin{aligned} f_{\bar{\mathbf{x}}}(t,s) &= \frac{1}{d}\sum_{i=1}^{d}\Big(x_{i1}\cos(\lambda_1 t)\cos(\mu_1 s) + x_{i2}\cos(\lambda_1 t)\sin(\mu_1 s) + x_{i3} sin(\lambda_1 t) + \ldots\Big) \\ &= \frac{1}{d}\sum_{i=1}^{d} f_{\mathbf{x}_i}(t,s) \end{aligned}$$

- *Linear relationships.* If a point $\mathbf{y}$ lies on a line joining $\mathbf{x}$ and $\mathbf{z}$, then for all values of t and s, $f_{\mathbf{y}}(t,s)$ is between $f_{\mathbf{x}}(t,s)$ and $f_{\mathbf{z}}(t,s)$.

  Since $\mathbf{y}$ lies on a line joining $\mathbf{x}$ and $\mathbf{z}$ we can express it as $\mathbf{y} = \mathbf{x}(1-l) + l\mathbf{z}$ for some $l \in (0,1) \cap \mathbb{R}$ and $f_{\mathbf{y}}(t,s)$ as

  $$f_{\mathbf{y}}(t,s) = \big(x_1(1-l) + lz_1\big)\cos(\lambda_1 t)\cos(\mu_1 s) + \big(x_2(1-l) + lz_2\big)\cos(\lambda_1 t)\sin(\mu_1 s) + \big(x_3(1-l) + lz_3\big)\sin(\mu_1 s) + \ldots$$

  and again applying the distributive and associative properties we have

  $$f_{\mathbf{y}}(t,s) = f_{\mathbf{x}}(t,s)(1-l) + l f_{\mathbf{z}}(t,s)$$

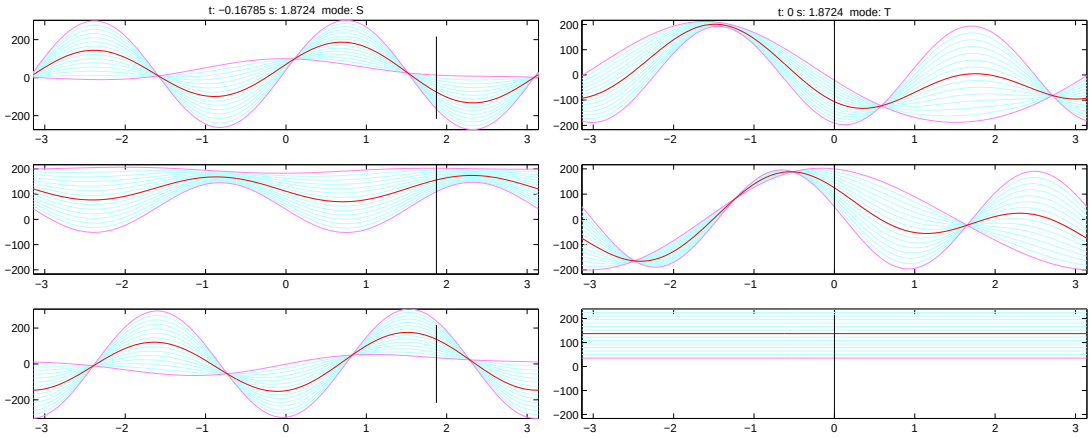


Figure 4.6: Surfaces for points (in cyan) equally spaces between two points (in pink). The surface associated to the mean appears in red.

4.4 Comparison with Wegman's Curves

In this section, we wish to compare Wegman's Algorithm with the derivative projections. Consider Wegman's curve for 3 dimensional data:

$$\mathbf{w} = \sqrt{\frac{2}{3}}\Big(\sin(\lambda_1 t), \cos(\lambda_1 t), \sin(\lambda_2 t)\Big)$$

Then this structure in 3 dimensional space is a cylinder such as shown in Figure 4.7 (the slight barrelling effect in that diagram is because we have introduced a normalization of the curve to unit length). Since the ends of the cylinder may be joined because of the periodicity of the trigonometric functions, a better description is a torus. Indeed any 3 dimensional projection of higher dimensional data which includes two terms of the form

$$\mathbf{w} = \sqrt{\frac{2}{3}}\Big(\sin(\lambda_i t), \cos(\lambda_i t), *\Big)$$

where $*$ is not a function of λ_i, will be a torus if $*$ is periodic. Therefore Wegman's Curves are projecting the data onto the surface of a torus. Of course not all projections of high dimensional data will show this structure. e.g. if we have $n \geq 5$,

$$\mathbf{w} = \sqrt{\frac{2}{n}}\Big(\sin(\lambda_i t), \cos(\lambda_j t), \sin(\lambda_k t)\Big)$$

for example, where $i \neq j, j \neq k, i \neq k$ will project the data onto a cube since all axes are orthogonal to one another.

However the derivative curves

$$\mathbf{w}_1 \propto \big(\cos(\lambda_1 t)\cos(\mu_1 s), \cos(\lambda_1 t)\sin(\mu_1 s), \sin(\lambda_1 t)\big) \tag{4.9}$$

form the surface of a sphere which is quite topologically distinct from the torus (e.g. the Euler characteristic of the sphere is 2 while that of the torus is 0). In the context of the current discussion, it is interesting to note that every closed curve on the surface of a sphere cuts the sphere into two parts while there exist many closed curves on the torus which do not have this property.

As with the torus, we may consider 3 dimensional projections onto the natural basis of higher dimensional data:

- 4 dimensional data will contain 4 natural projections. The first is (4.9) which shows a sphere. Another is

  $$\mathbf{w}_1 \propto \big(\cos(\lambda_1 t)\cos(\mu_1 s), \cos(\lambda_1 t)\sin(\mu_1 s), \cos(\lambda_2 t)\cos(\mu_2 s)\big) \tag{4.10}$$

  which will be a cylinder since the last term is orthogonal to the circle formed by the first two. The other two projections will also be cylinders of similar form to (4.10).

- 5 dimensional data contains $\binom{5}{3}$ i.e. 10 natural projections only the first of which (4.9) will be a sphere. Others such as

  $$\mathbf{w}_1 \propto \big(\cos(\lambda_i t)\cos(\mu_i s), \cos(\lambda_i t)\sin(\mu_i s), \cos(\lambda_j t)\cos(\mu_j s)\big) \tag{4.11}$$

  $i \neq j$ will be more difficult to analyze. Some of the most interesting are given by projections like

  $$\mathbf{w}_1 \propto \big(\cos(\lambda_i t)\cos(\mu_i s), \cos(\lambda_i t)\sin(\mu_i s), \sin(\lambda_j t)\big) \tag{4.12}$$

  which produces certain discontinuities.

- When the dimensionality is sufficiently high, we can have the cube projections exhibited by the Wegman's Curves e.g.

  $$\mathbf{w}_1 \propto \big(\cos(\lambda_i t)\cos(\mu_i s), \cos(\lambda_j t)\sin(\mu_j s), \cos(\lambda_k t)\cos(\mu_k s)\big) \tag{4.13}$$

where $i \neq j, j \neq k, i \neq k$.

Thus every high dimensional data set will be projected onto a high dimensional torus by Wegman's Curves. The same data sets will be projected by the derivative curves onto a shape which extends the torus by one dimension in some way: our mental image is of three dimensional "cross-sections" of the shape forming a sphere which is linked along its spine to other "cross-sections" (also spherical) by a higher dimensional cylinder. Alternatively, one can state that just as one can construct a torus from a square by joining opposite ends of the square to make a cylinder, and then bending the cylinder so that opposite ends can be joined again, we can make the current shape by joining opposite faces of a cube and then repeating this twice with the resulting shape.

4.4.1 Identifying Clusters

Consider the simple case

$$\mathbf{w}_1 = \big(\cos(\lambda_1 t)\cos(\mu_1 s), \cos(\lambda_1 t)\sin(\mu_1 s), \sin(\lambda_1 t)\big) \tag{4.14}$$

Then

$$\begin{aligned}
\frac{\partial \mathbf{w}_1}{\partial t} &= \big(-\lambda_1 \sin(\lambda_1 t)\cos(\mu_1 s), -\lambda_1 \sin(\lambda_1 t)\sin(\mu_1 s), \lambda_1 \cos(\lambda_1 t)\big) = -\lambda_1 \mathbf{w}_2 \\
\frac{\partial \mathbf{w}_1}{\partial s} &= \big(-\mu_1 \cos(\lambda_1 t)\sin(\mu_1 s), \mu_1 \cos(\lambda_1 t)\cos(\mu_1 s), 0\big) = -\mu_1 \cos(\lambda_1 t)\mathbf{w}_3
\end{aligned}$$

Then a cluster may be identified as consisting of data points which are such that the values of each coordinate is approximately the same for all data points. This will show up as a tight cluster of lines representing the projections of the data onto $\mathbf{w}_1$. Alternatively, there may be data sets in which the relative proportions of the various coordinates is the crucial factor: for example, a young organism may have values very far from the adult organism but nevertheless may have these values in similar proportions across all measurements. This type of clustering will show up as proximity in the second or third projections onto $\mathbf{w}_2$ or $\mathbf{w}_3$. This last type of clustering is somewhat difficult to identify in the projections onto $\mathbf{w}_1$ alone: this projection will parallel the projection of the cluster but some way off so that in even a moderate sized data set, it may become lost in the crowd of other projections. It will however be easy to spot in the projections onto $\mathbf{w}_2$ or $\mathbf{w}_3$.

Clearly this argument holds for higher dimensions of projection too since, in

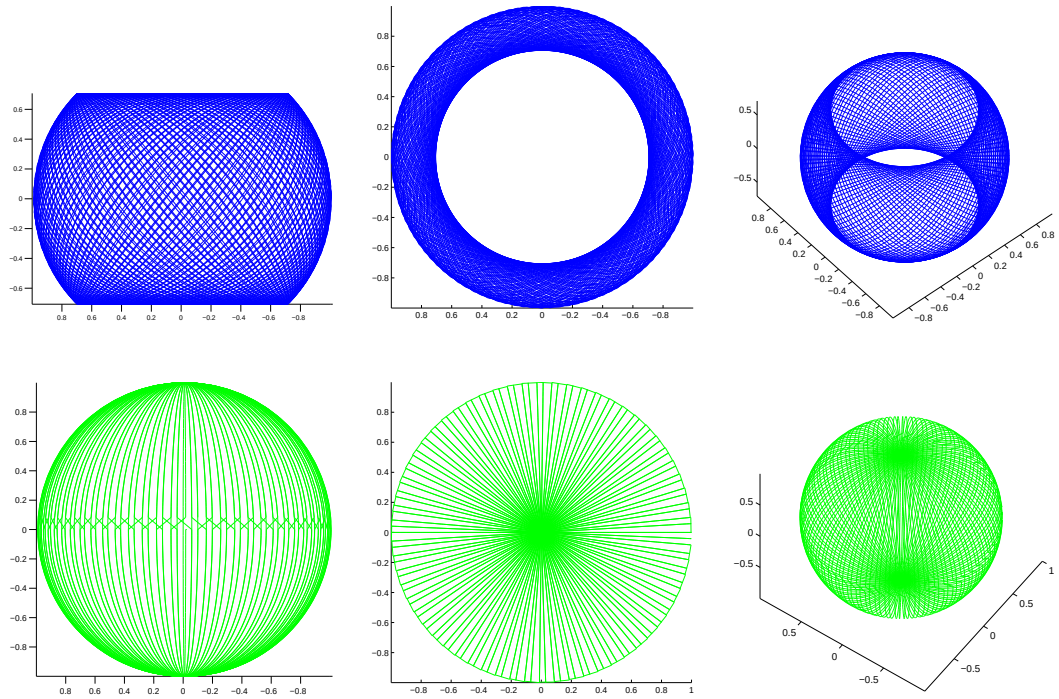


Figure 4.7: The top diagram shows three 2D projections of Wegman's Curves; the bottom diagram shows three 2D projections of the new curves presented in this chapter.

all cases, $\mathbf{w}_2 \propto \frac{\partial \mathbf{w}_1}{\partial t}$ and $\mathbf{w}_3 \propto \frac{\partial \mathbf{w}_1}{\partial s}$ and so, in general, clusters can be identified in the three projections in the same manner as described above.

4.4.2 Comparing the space filling property

Wegman [179] points out one advantage that his formulation has over Andrews' Curves is that the Wegman's Curves are more space filling in the sense that the distance between the projection of an arbitrary point in the projection space and the nearest Wegman's curve in that space is less than its distance to the nearest Andrews' curve. An indication of the way the curves lie on surfaces in the 3D sphere is shown in Figure 4.7. The top diagram shows three 2D projections of the Wegman's Curves; the bottom diagram shows three 2D projections from the new algorithm. We see that Wegman's Curves have a gap near the poles of the sphere whereas our projection method does not.

However when we examine the distances in other dimensions we see a different story emerging. Figure 4.8 shows the mean expected distances for various dimensionality of data (and hence sphere). To make the comparison fair we have

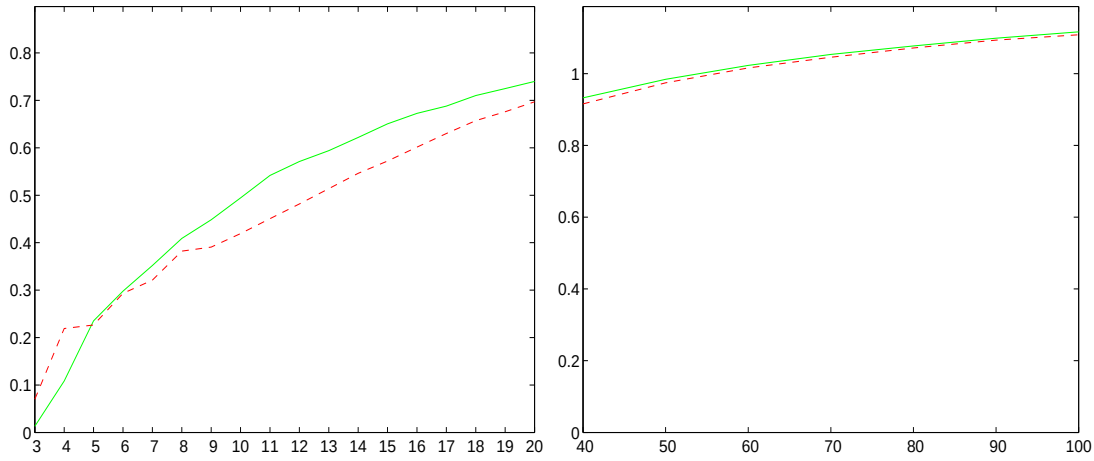


Figure 4.8: Average distances to the closest point on the surface/curve (or on its reflection through the origin). Red dashed: Wegman's Curves. Green solid: our surfaces.

used Wegman's Curves on the interval $[-63\pi, 63\pi]$ which is quantized in intervals of 0.1 to give 3959 curve points. Our curves are parameterized by t and s both drawn from $[-\pi, \pi]$ again quantized in intervals of 0.1 which gives $63^2 \approx 3959$ points. We used λ and μ values from 1 to $\lceil n/3 \rceil$ for our surfaces i.e. λ_i and μ_i values were consecutive integers. In the Wegman's Curves, we used as lambda coefficients the square roots of the primes numbers: $\sqrt{2}, \sqrt{3}, \sqrt{5}, \sqrt{7}, \ldots$, (assuring in that way the linear independency over the rationals). We see that generally Wegman's Curves are no worse than those of the current algorithm and indeed often slightly better.

If we renounce the distance preservation property of our surfaces, and we use also linearly independent coefficients for the μ and λ coefficients, our surfaces are as space filling as the Wegman's Curves for some values of μ and λ. In Figure 4.9 we can see the result of using the values: $\lambda_1 = \mu_1 = 2^{\sqrt{2}}, \lambda_2 = \mu_2 = 3^{\sqrt{2}}, \lambda_3 = \mu_3 = 5^{\sqrt{2}}, \lambda_4 = \mu_4 = 7^{\sqrt{2}}, \ldots$, i.e. the prime numbers raised to the root square of two. Figure 4.10 shows the results when the used values are: $\pi+1, \pi+2, \pi+3, \pi+4, \ldots$.

In the first case, in which the derivative surfaces are a bit less space filling than Wegman's Curves, they have the advantage of having the distance preservation property and providing a more convenient way of moving through the different projections changing the values of t and s, giving us a more orderly way to surf the different projections, as we will see in the next section.

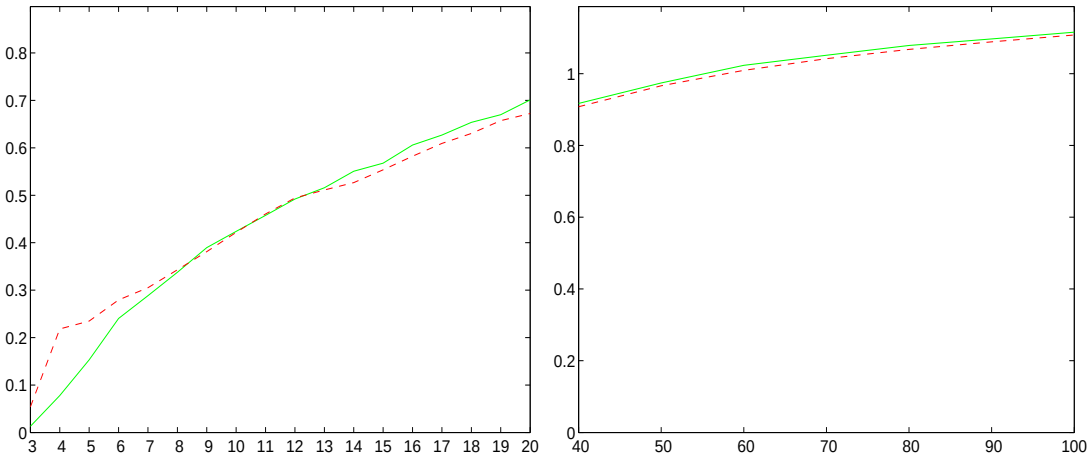


Figure 4.9: Average distances to closest point on surface/curve using λ and μ values obtained from the sequence: $(2^{\sqrt{2}}, 3^{\sqrt{2}}, 5^{\sqrt{2}}, 7^{\sqrt{2}}, 11^{\sqrt{2}}, \ldots)$. Red dashed: Wegman's curves. Green solid: our surfaces.

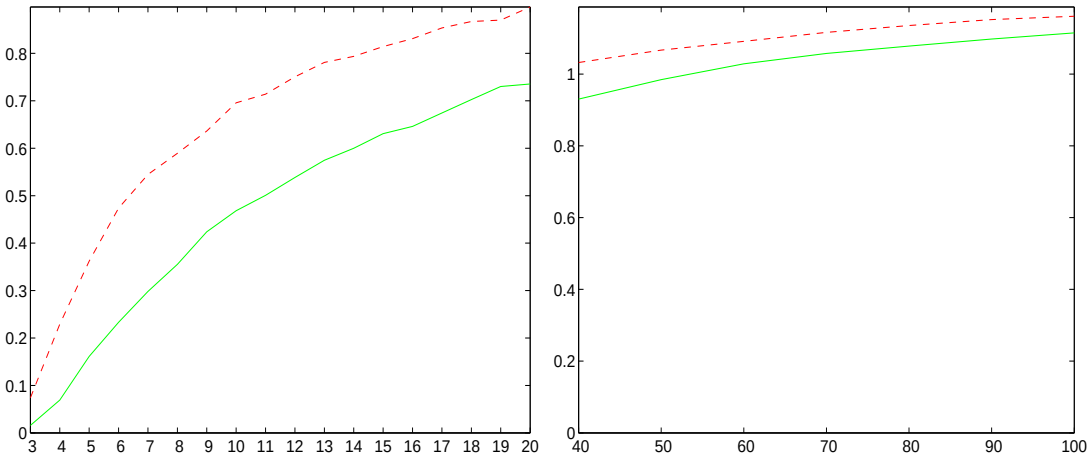


Figure 4.10: Average distances to closest point on surface/curve using λ and μ values obtained from the sequence: $(\pi + 1, \pi + 2, \pi + 3, \pi + 4, \ldots)$. Red dashed: Wegman's curves. Green solid: our surfaces.

4.5 Data Exploration with Curves

Now we have two representations of the data: we have three curves (see Figure 4.11) which are a simple extension of Andrews' Curves which we saw earlier; but we also have a representation of each data point as a point in three dimensional space as in Figure 4.15 — it is difficult to convey the visual clustering effect of this representation as we move along either curve in a static diagram but the impression of clusters in the dynamically changing environment is very strong.

This suggests the following interactive process for identifying clusters in data: calculate the curve for each data point (in terms of parameters s and t) and search for clusters of points which are performing similarly over at least a local part of the curve and which are also distinct from other points' curves over the same part of the curve. We will initialize our curves with $s = 0$ and $t = 0$ and then simply progress along the curves looking for small sections in which such a group of points can be identified. Of course such a group will not remain distinct from the other curves throughout all its length but, if it is to qualify as a group, it must remain coherent, forming a small bundle of curves through all values of s and t. When we identify such a group, we will remove it from the data set and then repeat the process with the remaining data points.

Since the method is interactive, we are actually happy to work with two types of displays of the data: the first is that described above and shown in Figure 4.11; the second is the view in 3 dimensions of the data moving in space. It is difficult to do justice to this second view on a static page but the impression of clusters of objects moving together is very strong in this display.

The number of samples we are going to work with is not large (118 samples). One of the criticisms that Andrews himself makes of his curves [2] is that the curves are only useful when the number of points is tiny. Actually the Andrews' Curves and the variant that we propose in this paper can be used without major problems with a greater number of points if the curves are combined with a brushing mechanism [8] that allows us to highlight those points/curves (depending on which representation, Figure 4.11 or Figure 4.15 which we want to check) to see if they constitute a cluster (of course when Andrews wrote his paper computing facilities were much less powerful than those we have now). In our implementation, we can instantly highlight a group by changing the colour of the curves of its members and the corresponding points in 3D space using point and click operations (this is known as brushing). As noted above, when we are convinced

that such a group is moving coherently through all values of s and t, we remove them from the display and continue our search for new clusters. Finally, as we will see later, sometimes the initial cluster can be refined by subdividing it into several sub-clusters.

The tool that we have developed to help us in the process of classification is a MATLAB prototype program that lets the user visualize interactively different slices of our surface. The two directions of the surface slices are perpendicular to each other; we call the slices where s changes, S-slices which are thus slices which retain some specific but constant t value; similarly we call T-slices those slices where t changes, and in which s is constant. A vertical line indicates the point where the current point representation is taken from. It is possible to change between the two types of slices, and also it is possible to move interactively changing the s value of a T-slice, or the t value of a S-slice (that is, changing the point where we cut the surface to obtain the slices); this gives a kind of grand tour [181] that is multidimensional, since in the slices we have a simultaneous view of all the projections in the range $[-\pi, \pi]$.

Finally, it is also possible to display the two-dimensional projection of a three-dimensional grand tour that can be defined using these surfaces. In this grand tour, the user can decide to move in a direction where the s value changes or in a direction where the t value changes. One possible extension is to give the user the ability to interactively change the μ and λ values. In this grand tour it is also possible to select the points in which we are interested.

4.5.1 The algae data set

As we discuss in the previous chapter, the data set comprises data from 118 samples of algae; some of the algae have been classified (by humans) into classes labelled 1 to 9 while the remainder have been left unlabelled (we give them a label 0). Some of the label 0 algae may be members of classes 1 to 9 while others may be a totally different type of algae. The data is 18 dimensional where a typical dimension gives the quantity of pigment of a specific type which has been measured in the sample. We have pre-processed the algae data set by centering the samples, i.e. subtracting the mean of all samples from each sample.

This data set has the advantage that we can check if algae which have been identified as belonging to one group do, in fact, remain together throughout multiple values of the parameters (see below) while simultaneously investigating

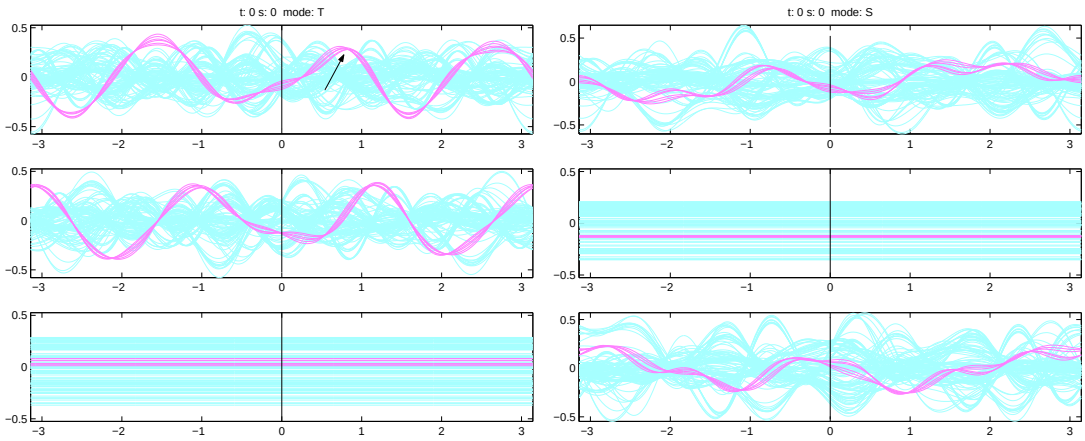


Figure 4.11: Left: the initial T-slices (we have brushed some curves). Right: the initial S-slices (with the same highlighted curves).

whether the algae labelled 0 are possibly members of existing groups or come from quite distinct groups of algae.

4.5.2 Initializing the clustering

Initially our tool shows the projection and the slices corresponding to the values $s = 0$ and $t = 0$. We can make an initial decision to concentrate on one of the slices and then (either by watching the evolution of the grand tour or by inspecting different slices of our surfaces) check if we can identify selected points which fall within the same cluster.

4.5.3 First cluster

For the algae data set, we start by focusing on the first T-slice. Near the value $t = 0.8$ appears a potential candidate cluster. As one can see in Figure 4.11, there is a very strong indication at that point that a cluster can be identified. We are able to confirm visually that the band we have brushed in these figures remains coherent throughout the s, t-space and so we conclude that we have identified a cluster of algae with similar properties.

Indeed, since we have class labels, we are able to verify that these points are within class number 8 of the original data set. Therefore, we delete these points and their associated curves and continue the exploration.

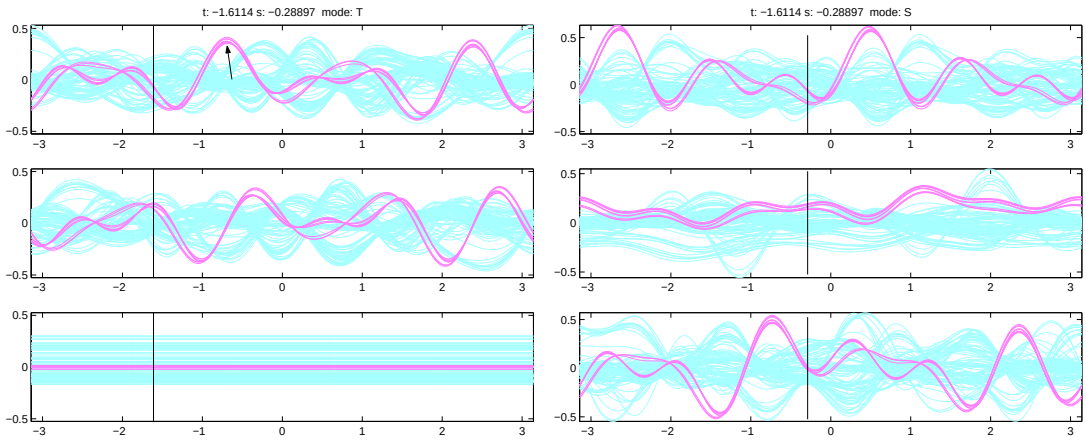


Figure 4.12: Left: the T-slices at $s = -0.28897$ (some curves are highlighted). Right: the S-slices at $t = -1.6114$ (with the same highlighted curves).

4.5.4 Second and third cluster

Now we can notice that around the value $t = -0.7$ in the first T-slice for $s = -0.28897$, there exists another possible cluster. After highlighting the curves and analyzing their evolution over different parts of our surfaces (Figure 4.12), we conclude that is possible to make a subdivision of this cluster into two distinct clusters, as one can see in the slices of Figure 4.13. Again checking the existing classification of the data reveals that this cluster corresponds to the classes 7 and 9. The proximity of the clusters suggests some kind of relation between the two groups of algae. These points and their associated curves are then removed from the display.

4.5.5 Fourth cluster

In the third S-slice (that does not depend on the value of t), around the value $s = -2.5$ one can clearly identify another candidate cluster and indeed, we find that all the selected points belong to class 4. After investigating other slices it was decided to include in the cluster an additional point. As one can see in Figure 4.14, this inclusion is quite reasonable since the shape of the curve is very similar to the other curves in the cluster. This is an interesting finding, since this curve corresponds to a point that is not classified in the original data set (i.e. is labelled 0). The projection shown in Figure 4.15 reinforces the conclusion that we have identified a previously unclassified data point and have been able to classify it as class 4 with a reasonable degree of confidence.

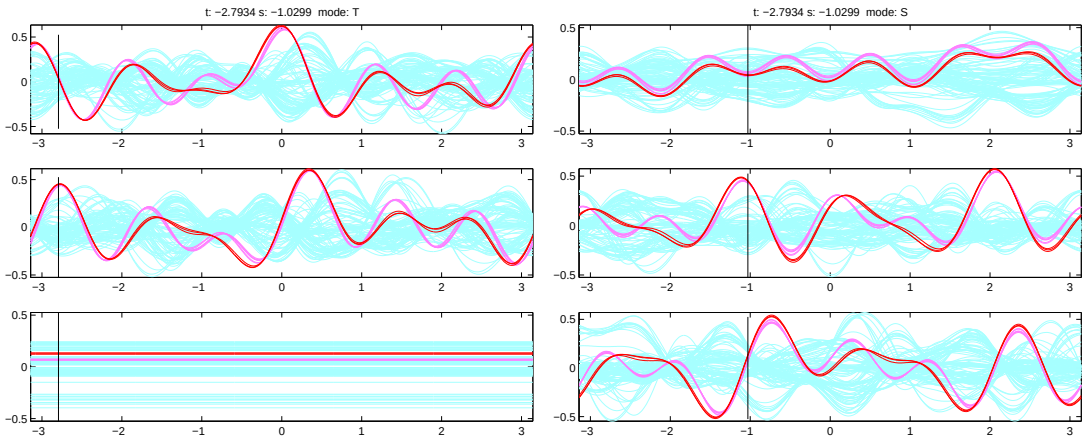


Figure 4.13: Left: the T-slices at $s = -1.0299$ (two different sub clusters has been identified). Right: the S-slices at $t = -2.7934$ (the same two different sub clusters).

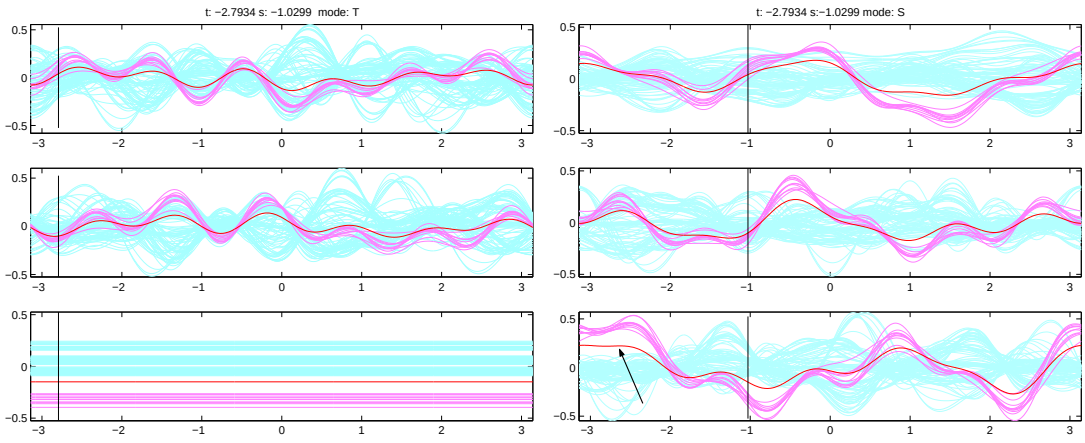


Figure 4.14: The initial highlighted curves of cluster four, plus one additional curve with a similar behaviour. Left: the T-slices at $s = -1.0299$. Right: the S-slices at $t = -2.7934$ (the arrow indicates the place which motivates the initial selection).

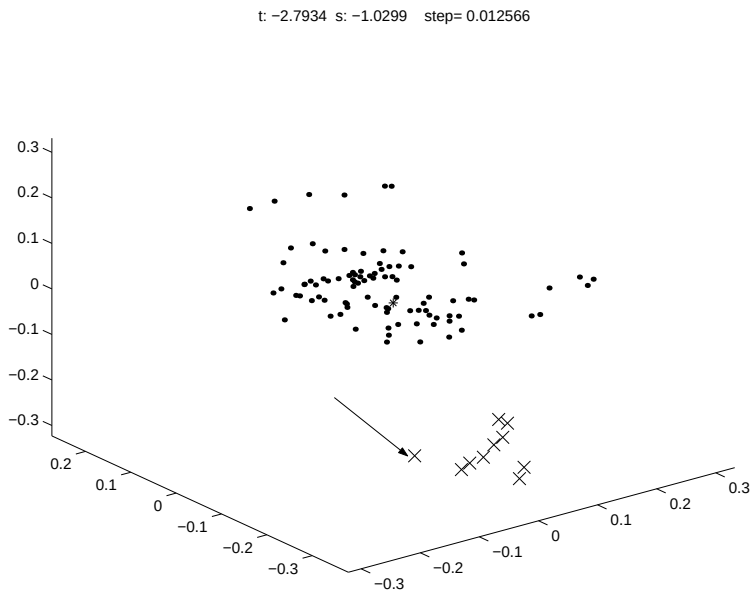


Figure 4.15: The projection obtained at $t = -2.7934$ $s = -1.0299$ (the arrow points the additional included point, the highlighted points are represented using crosses).

4.5.6 Fifth cluster

If we watch the second T-slice around the value $t = 1.43$, again we can observe a candidate cluster which we may highlight and check that, although not as clear as in the previous cluster (the points of this cluster are more dispersed), the curves have a very similar shape which suggests that they form a cluster. On checking we confirm that the curves correspond to points in class 6. What is more, if we include the next closest curve (which corresponds to a previously unclassified point) we can see that its behaviour suggests that we have identified another point in the class (Figure 4.16).

4.5.7 Sixth cluster

Now the third S-slice around $s = 0$ gives a clue to unveiling another candidate cluster; highlighting the curves reveals the similarity in the curves, although some of them have a more erratic behaviour than that seen in previous groups. Performing a selection of curves around $s = 1.5$, it was possible to eliminate from this candidate cluster two of the most erratic curves (Figure 4.17). Subsequent analysis shows that in the cluster there are points from the classes 1 and 5, and also unclassified points. As one can see in Figure 4.18 (only the points from

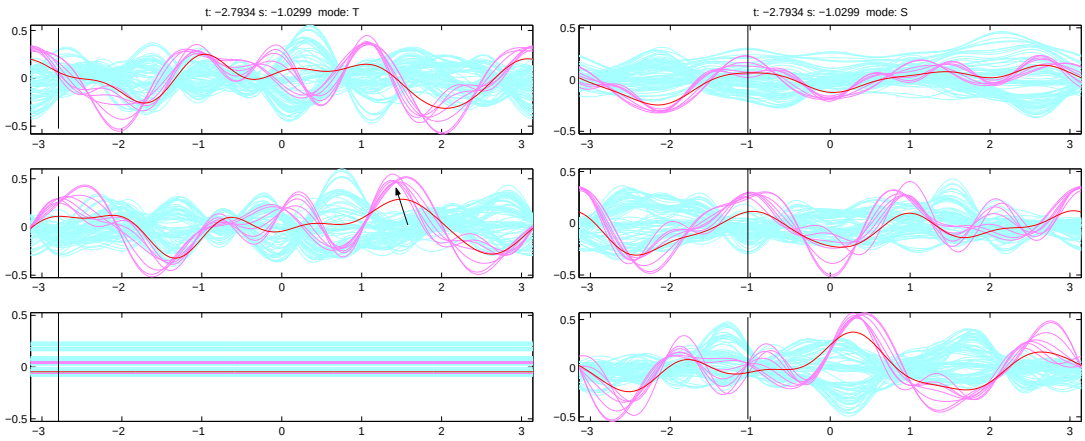


Figure 4.16: The initial highlighted curves of cluster five, plus the next closest curve. Left: the T-slices at $s = -1.0299$ (the arrow indicates the place which motivates the initial selection). Right: the S-slices at $t = -2.7934$.

classes 1 and 5 are shown) the two classes are very close, and it is impossible to subdivide them which explains the difficulty in clustering. Still, it is worth noting that no point from the original classes 1 and 5 was outside the identified cluster and that it was possible to group some of the unclassified points.

4.5.8 Seventh cluster

Analyzing the behaviour of the projection of the grand tour for the remainder of the data set, it was possible to identify another cluster; in Figure 4.19 one can see two of these projections. One of the points exhibits a behaviour a bit different from the rest, but globally that point is following the movement of the rest of the points (Figure 4.20). Again we have been able to reveal an interesting fact, since all the points in this cluster were unclassified in the original data set. However, this time we can not label the cluster as being one of the original classes. Thus we feel emboldened to suggest that we have uncovered a class of algae which was not previously designated as a class in the original data analysis.

4.5.9 Eighth cluster

Watching the evolution of the grand tour, two projections were obtained that let us identify another cluster (Figure 4.21). Indeed, from our observation of the highlighted curves, we conclude that is possible to perform a subsequent subdivision (Figure 4.22). This cluster is quite interesting because it is made up

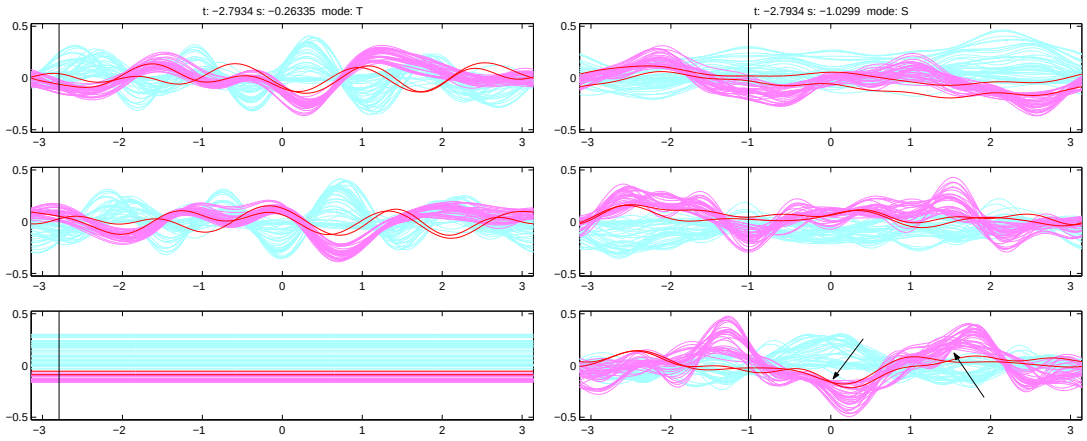


Figure 4.17: The refined cluster six, plus some curves in the initial selection. Left: the T-slices at $s = -0.2633$. Right: the S-slices at $t = -2.7934$ (the arrows indicate the place which motivates the initial selection and the place that allows us to refine the selection).

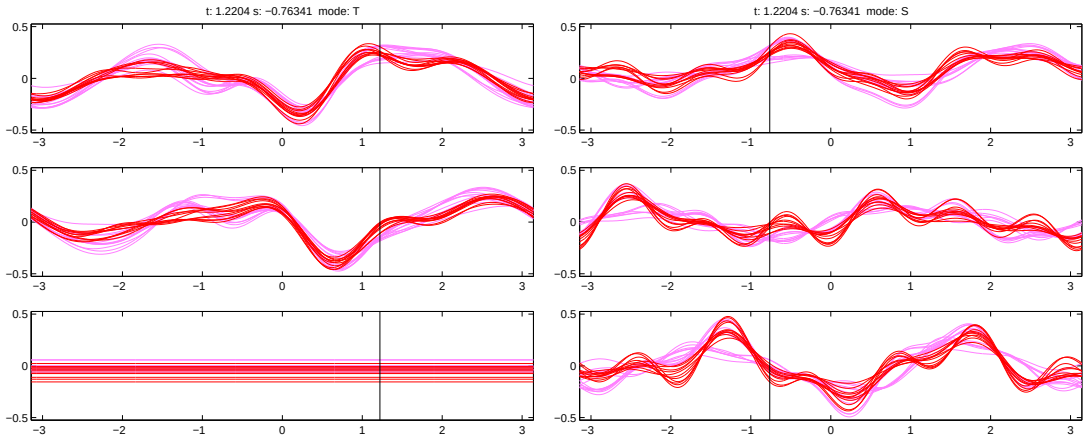


Figure 4.18: The curves for the classes 1 and 5. Left: the T-slices at $s = -0.76341$. Right: the S-slices at $t = 1.2204$.

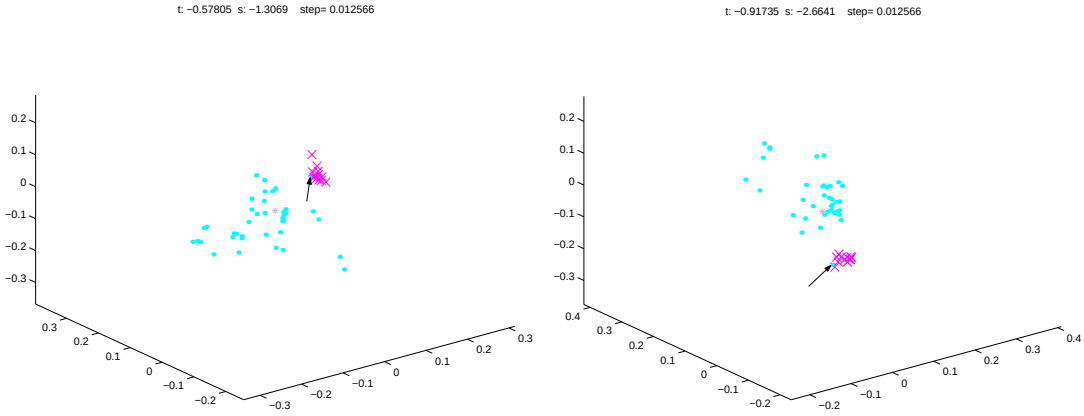


Figure 4.19: Two different projections of the grand tour (the arrow points to the point with a bit different behaviour). Left: $t = -0.57805, s = -1.3069$. Right: $t = -0.91735, s = -2.6641$.

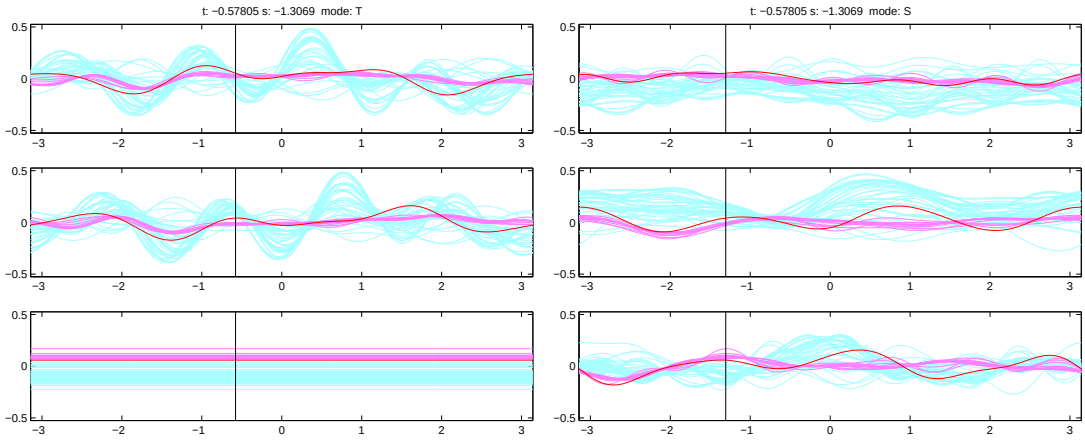


Figure 4.20: The curves in the cluster seven, the darker one shows the different behaviour of one of the points/curves. Left: T-slices at $s = -1.3069$. Right: S-slices at $t = -0.57805$.

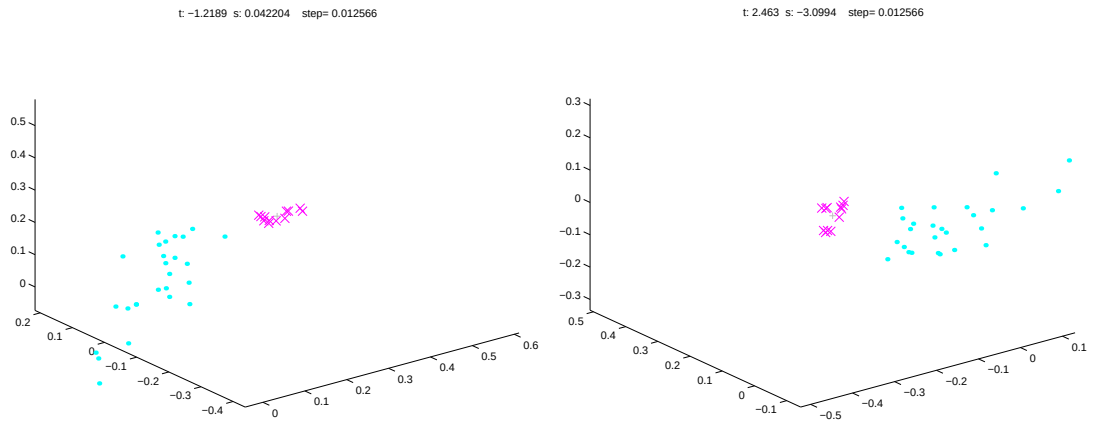


Figure 4.21: Two different projections of the grand tour where the points in the cluster eight have been highlighted. Left: $t = -1.2189, s = 0.042204$. Right: $t = 2.463, s = -3.0994$.

of all the points of class 3, but also includes one of the unclassified points and we have been able to identify a division of this class into two sub-clusters.

4.5.10 Ninth cluster

Again from the grand tour, a possible cluster was found; after highlighting the curves and watching other slices of our surface, we found that all points but one give curves with similar shape and which remain fairly close to one another throughout the tour (Figure 4.23). In the identified cluster, all the points are from class 2, and, in addition, we have been able to classify 6 additional unclassified points.

4.5.11 The remaining points

Within the remaining points, it was possible to identify another three curves with a behaviour that gives us cause to believe that they also form a cluster (Figure 4.24). The rest of the (eleven) curves do not have a structure which suggests that they belong to no discernable grouping in this data set.

We have thus gone some way to differentiating the various classes in this data set.

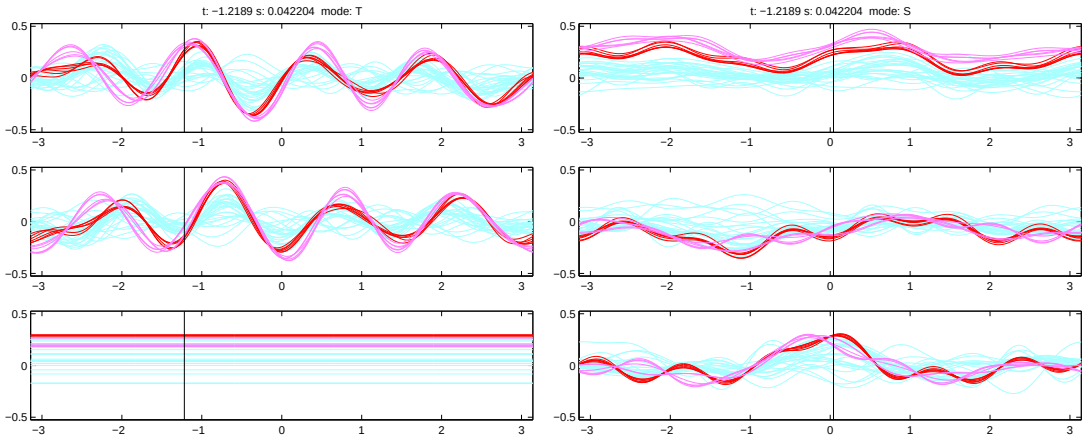


Figure 4.22: The curves in the cluster eight (the two sub cluster are shown in different colours). Left: T-slices at $s = 0.042204$. Right: S-slices at $t = -1.2189$.

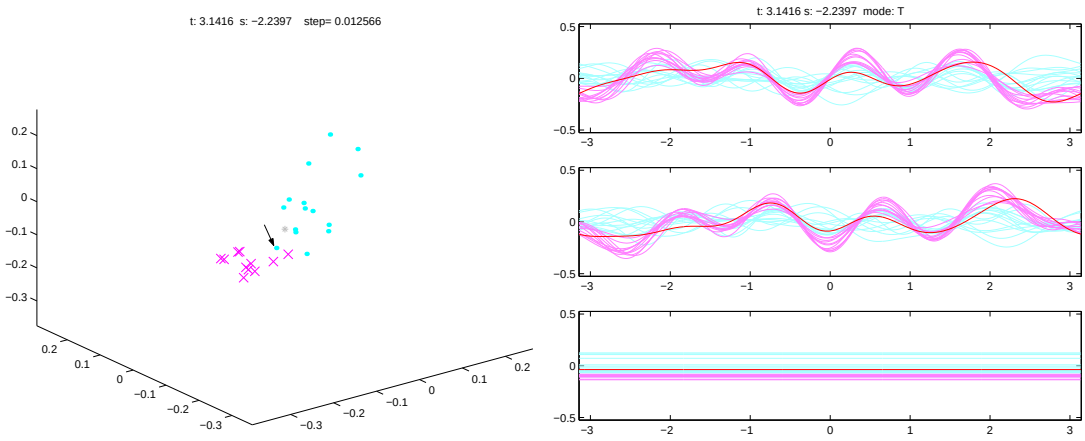


Figure 4.23: The points/curves in the cluster nine. Left: Projection of the grand tour at $t = 3.1416, s = -2.2397$ (the arrow points to the point a bit different from the rest). Right: T-slices at $s = -2.2397$ (the darker curve is a bit different from the rest).

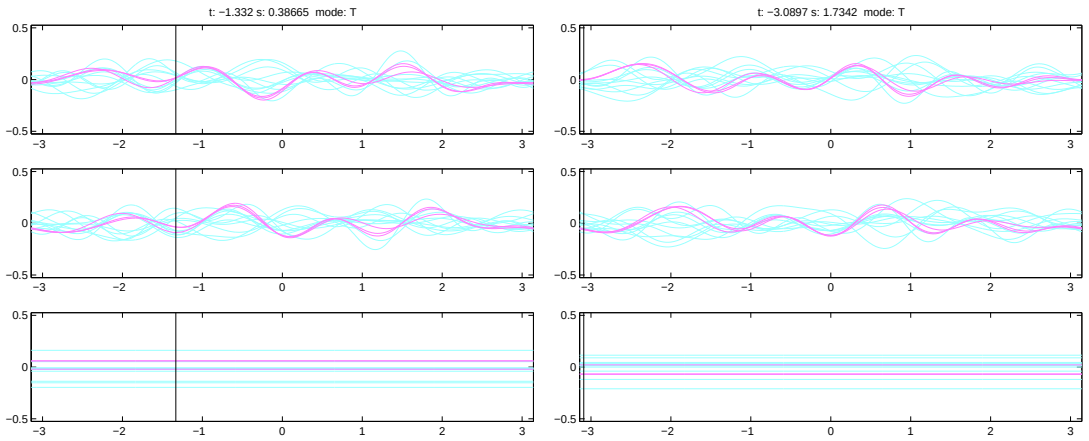


Figure 4.24: Left: T-slices at $s = 0.38665$. Right: T-slices at $s = 1.7342$.

4.6 Conclusions

We have discussed two very simple extensions to an existing, indeed, rather old technique for exploratory data analysis. The advance of computational power has made possible extensions which Andrews could only dream about thirty years ago.

We have enumerated and proved some of the properties that the derivative projections hold. We have compared the derivative projection with Wegman's Curves to conclude that although not so space filling as Wegman's Curves in some circumstances, our surfaces are a very convenient way to order different projections of high dimensional data sets, enabling us to perform a well-organized exploration of the data set.

We have shown that the derivative projections can be used in two quite different but complementary ways:

1. The first allows us to walk along the curves using either of two parameters and find groups of curves which remain as a group for all possible values of the parameters.

2. The second allows us to use the human facility of identifying structure in moving three dimensional displays, something for which our evolution in a three dimensional visual environment has created excellent pattern matchers.

The combined use of the new display with techniques such as brushing and

linking[2], have let us identify most of the labelled clusters of a real data set, but also allowed us to classify some of the unlabelled samples, and also to discover new clusters and identify sub-clusters in one labelled cluster.

Apart from the advantage of providing us with a three dimensional projection of high dimensional data, the derivative representation has interesting theoretical properties which we have investigated.

In the next chapter, we give more applications of the new display. We combine it with the SOM to obtain a symbiosis that is beneficial for both the SOM and the derivative surfaces. We also investigate the use of colours to improve visualization.

[2]A mechanism for relating information in one plot to the information in another.

Chapter 5

Combining Visualization Techniques

In the previous chapter we introduced a new data analysis tool based on Andrews' curves and showed how to use it to find clusters in a real data set. Here we extend that tool and investigate its use in combination with the SOM.

The use of self-organizing maps to analyze data often depends on finding effective methods to visualize the SOM's structure. We propose a new way to perform that visualization using our surfaces. Also we show that the interaction between these two methods allows us to find sub-clusters within identified clusters. Perhaps more importantly, using the SOM to pre-process data by identifying gross features enables us to use Andrews' curves and our variant on data sets which would have previously been too large for the methodology.

We also show how to combine in the same representation the information obtained from two different views of the data (for example, raw data and a principal component projection of the data) obtaining a more robust exploratory data analysis tool.

5.1 The Combined Use of Self-Organizing Maps and Andrews' Curves

A Self Organizing Map (SOM) is a well known method for clustering data: it clusters in such a way that data points which are alike in some way, tend to be matched to neurons which are alike in some way and only such data points are so matched. We have seen the details of how it works in Section 2.3.4. After

training the SOM, the identification of clusters is not very difficult if we use an appropriate way to visualize the structure that the SOM has acquired. There are several ways to perform this visualization; in [170], a summary of such methods is presented. The visualization of the SOM enables us to perform an interactive type of data mining that has proven to be useful in exploratory data analysis [169, 120, 94, 35]

In this section we analyze the use of Andrews' curves as a visualization tool for the SOM's structure. We can use Andrews' curves to visualize the model vectors or centres of the SOM, that in general are multidimensional and so difficult to comprehend by inspection. We can use Andrews' curves to identify subclusters within the data set projected to a neuron of the SOM. Finally, we can use Andrews' curves to cluster the SOM, that is to discover groups of model vectors that form clusters; and to construct dynamic projections of the SOM grid.

On the other hand, we can improve the use of Andrews' curves if we reduce the number of points used to draw them; that is, instead of drawing the curves corresponding to all the points of a data set, we can train a SOM and use the Andrews' curves of the model vectors or centres to discover the general structure of the data set. Thus these two methods complement each other.

5.1.1 Using Andrews' Curves to visualize the structure of the SOM

Visualization of prototype vectors

The first application of the Andrews' curves to visualize the SOM is quite obvious. The *SOM Toolbox*, a software package for the Matlab computing environment[1], provides three different ways of visualizing the map prototype vectors: as line graphs, as bar charts and as pie charts. We have extended these visualization tools with a fourth one that uses Andrews' curves to visualize the prototype vectors. In Figure 5.1, it is possible to see the four visualizations of a SOM trained with the iris data set. We see that this first obvious application gains us little that we do not get from other methods and we do not pursue this method.

[1]The SOM Toolbox is available for free from `http://www.cis.hut.fi/projects/somtoolbox/`.

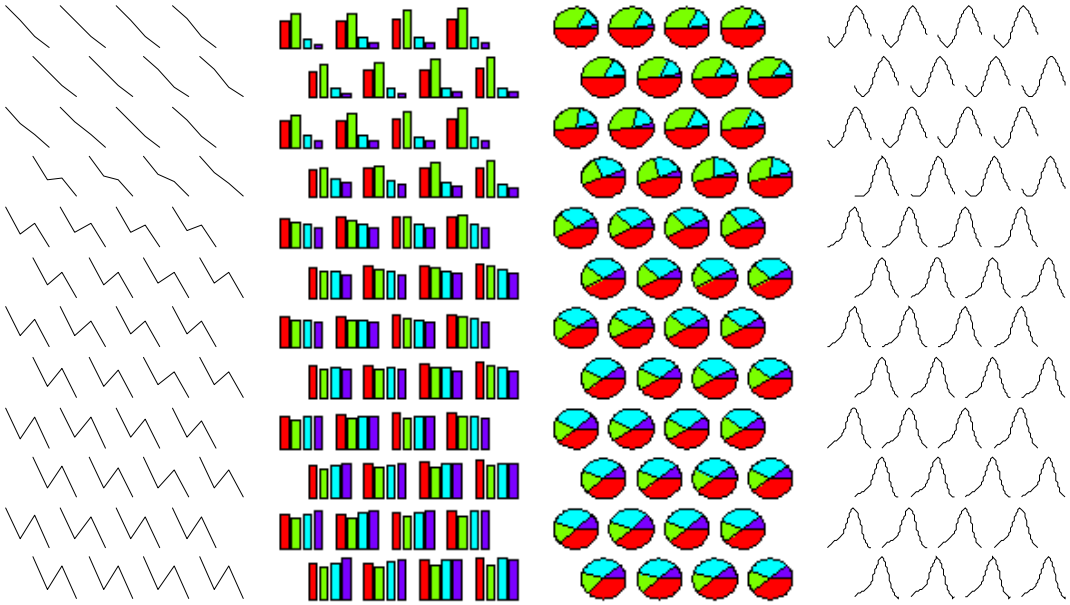


Figure 5.1: Different ways of visualizing the model vectors. The first three are in the existing literature. The fourth uses Andrews' curves. All are representing the model vectors of a SOM trained with the iris data set.

Finding sub-clusters

If the data set we are using has many clusters or if the dimensionality of the SOM in output space is not big enough, it may happen that after the training phase, one or several neurons were the BMUs (*Best Matching Units*) for data points belonging to different clusters: there is not a 1-1 mapping between the centres of the SOM and the clusters. Actually we do not require a 1-1 mapping but only that each centre identifies a separate class. The problem in which we are interested is the case when the SOM has not differentiated between the two clusters - the SOM has identified them as only one. However we can use Andrews' curves to identify the sub-clusters: the SOM has performed a crude classification perhaps finding some classes precisely but leaving others with some residual uncertainty.

To illustrate this we use a twenty dimensional artificial data set with 12 clusters (each of 85 points) and a SOM of dimensionality 3×3. Since the SOM has less neurons than there are clusters in the data set, it is inevitable that some neurons form the BMU of points from more than one cluster. We create the data set placing the centres of the 12 clusters on one of 12 orthogonal axes, then we add Gaussian noise in the other 8 axis and all the points were rotated using a randomly generated rotation matrix. The PCA projection of the data set can be

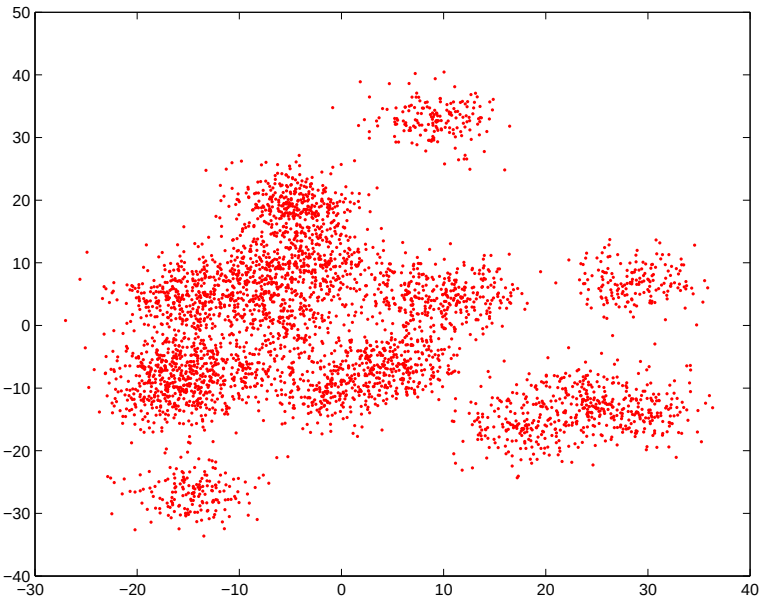


Figure 5.2: PCA projection of the artificial data set. Three cluster are easily identifiable, but the other 9 appear within two mixed clusters.

seen in Figure 5.2.

As we can see in Figure 5.3, the SOM identifies correctly four of the clusters. But there are also two neurons that are the BMUs for two different clusters, and one neuron that is the BMU for four different clusters. We will illustrate the use of Andrews' curves to disambiguate with this last neuron. If we use all the points for which that neuron is the BMU and we draw the Andrews' curves (in this case we are using two slices of our surfaces) we can identify clearly the four clusters as we can see in Figure 5.4. If we were to use all the data points in the original data set, we would have a far less clear picture. Thus we can use Andrews' curves to disambiguate information which contains residual uncertainty from the SOM clusters.

Clustering and projecting the SOM

There has been recent interest in using the SOM as an abstraction tool that enables us to substitute the input vectors with the model vectors as prototype vectors, and then use these to identify clusters [171]. In this way we can reduce the time needed by the clustering algorithm, since we have reduced the number of vectors to consider.

In the previous section we used Andrews' curves to draw the data points and

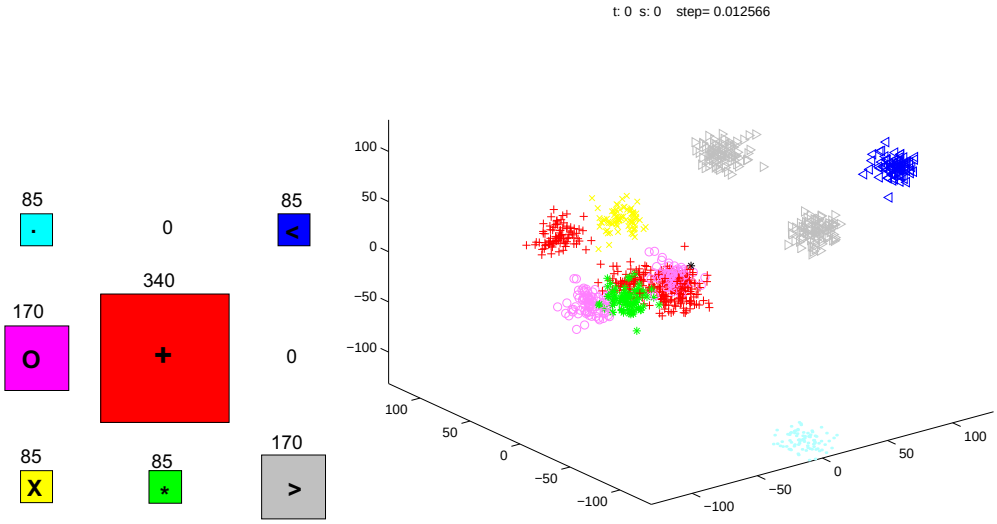


Figure 5.3: Left: the number of times each neuron is the BMU. Right: a projection of the data set using our surfaces.

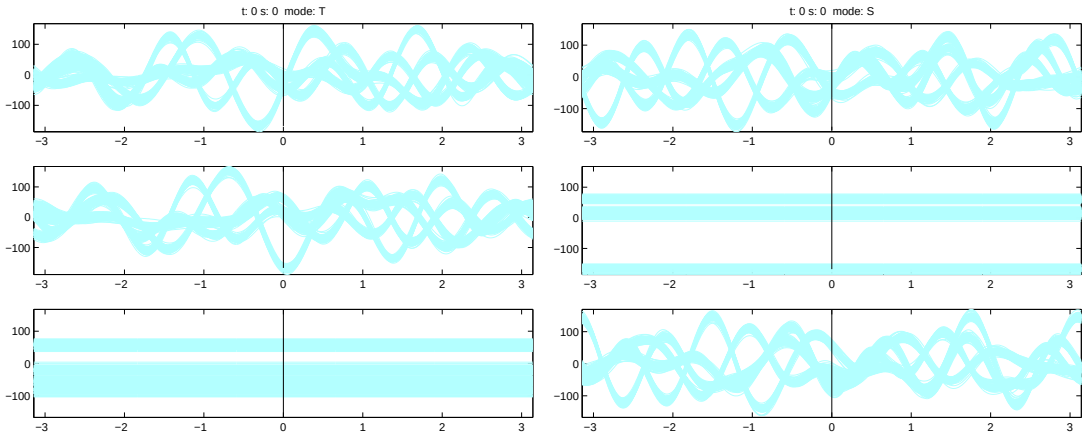


Figure 5.4: Left: T-slice when $s = 0$. Right: S-slice when $t = 0$. Individual sub-clusters can be identified within each plot.

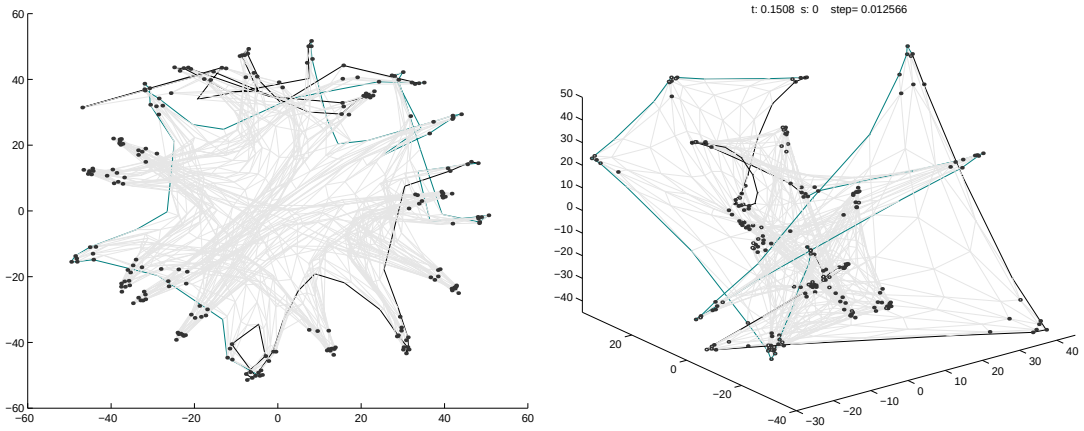


Figure 5.5: Left: Sammon Mapping of the prototype vectors of a SOM. Right: a snapshot of the grand tour of the Andrews' projections of the same SOM.

try to discover sub-clusters. We can also use Andrews' curves to draw and project the prototype vectors. We can discover the clusters, either using the static view of the slices of our surfaces, or using the evolution of the projected vectors in the pseudo grand tour.

The dynamic projection is interesting on its own. It is common to use projections such as the Sammon mapping [140] to try to visualize the structure of the SOM. But these kind of projections are static. With the use of Andrews' curves, we can obtain a dynamic three dimensional projection and see its evolution as we change the values of s or t. The evolution of the pseudo grand tour gives us an extra dimensional perception of the projection — we can feel the fourth (time) dimension as we allow the simulation to proceed. In Figure 5.5 we show the Sammon Mapping and one snapshot of the pseudo grand tour ; it has to be admitted than the inevitably static projection (right diagram) as seen on the printed page does not do justice to the clarity with which the clusters jump out on the computer monitor. The SOM was trained with an artificial data set consisting of 23 clusters with centres at the vertices lying on the axis of a 23-dimensional hypercube (and so the centres are of the form (30, 0, ..., 0), (0, 30, 0, ..., 0) etc.) plus another 52 additional dimensions with uniform (between 0 and 18) values to get a 75-dimensional data set with 4186 observations. To get the individual data points, we add zero mean Gaussian noise of variance 3 to each centre, but capped the noise at 2*variance in each dimension. This data set is then rotated in the 75-dimensional space with a random rotation matrix.

5.1.2 Enhancing Andrews' Curves using the SOM

Originally Andrews' curves were used to represent no more than a few tens of observations. With techniques such as brushing [8], (identifying a group of evolving structures and highlighting them) we can increase this number because we can select and highlight the curves in which we are interested (for example to determine the existence of clustering).

Using the SOM we can increase even more the number of curves. The idea is to use a SOM as an interface in which we select the regions in output space (the regular grid or line) which we want to draw. That is, we select one or more neurons to draw the data points for which these neurons are the BMUs. We can perform an analysis only with these curves and then continue by choosing another set of BMUs, perhaps with some BMUs in common with the first set, allowing in this way the connected study of all the points — we are walking through the data set using the interaction between the SOM, the Andrews' curves and the human operator to identify structure in the data. The SOM lets us navigate interactively through the data set keeping the number of curves to a manageable size.

A last improvement is the use of colour and linking [95, 79]. We can spread a colour map over a SOM and then associate a colour with the neurons, and similarly with the prototype vector and with the input vectors for which that neuron is a BMU. In this way we can use the clustering performed by the SOM as an extra information source. In Figure 5.6 we can see the slices of the surface obtained using the prototype vectors obtained by training the SOM with the same data set as in the previous section; the curves associated with dead neurons are not shown.

5.2 Combining Grand Tours

As noted by Embrechts and Herzberg [42], Andrews' curves suffer from strong dependence on the order of the variables; they propose trying different orders and scales of variables. We have ended the previous section suggesting the use of colour as a way to link the SOM with Andrews' curves. Here we use the same idea to combine two different groups of curves (obtained, for example, with different order of the variables) in only one display.

For each value of t and s our method gives three different values for each observation in the data set. Until now, we have used these values as coordinates

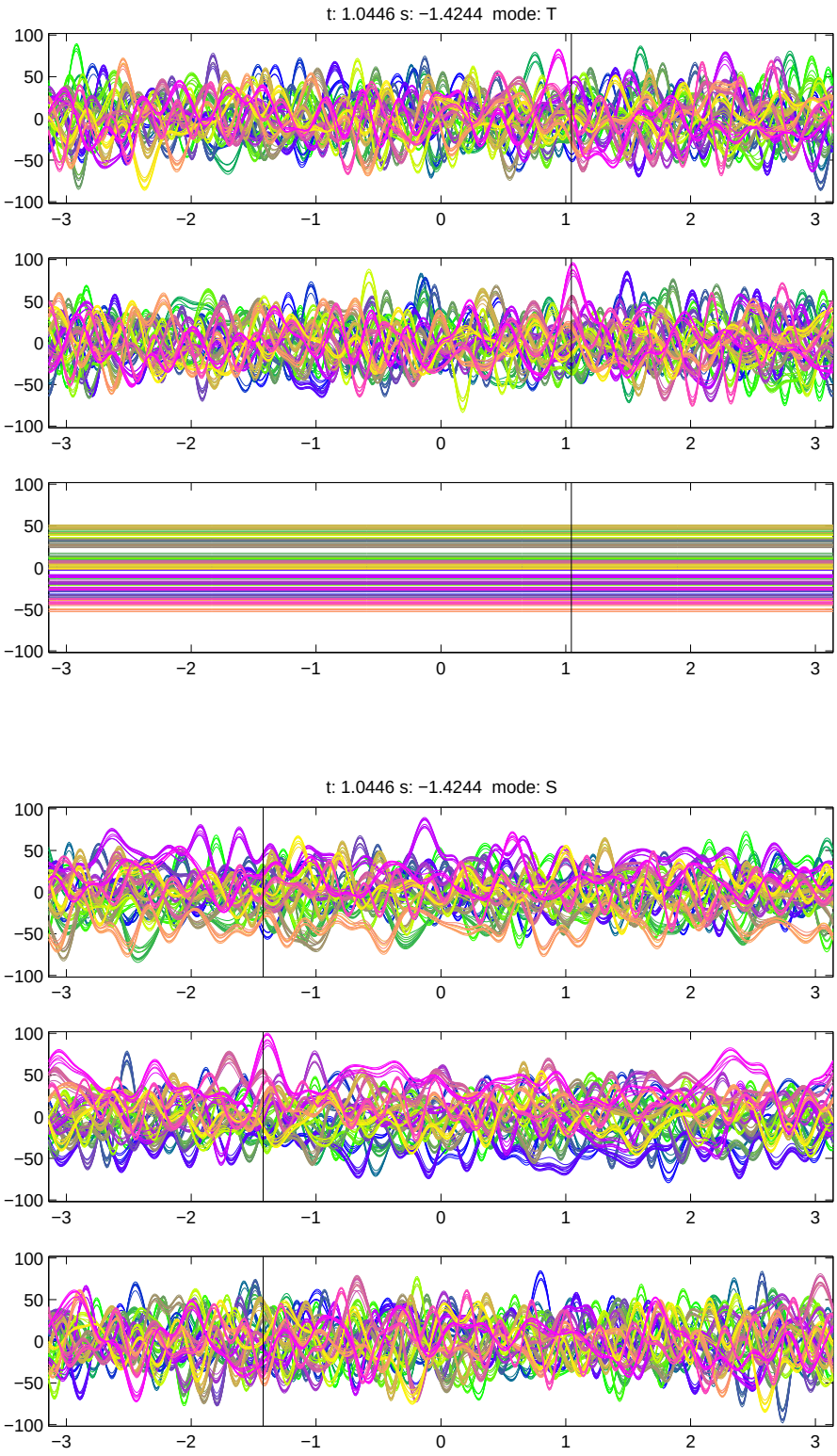


Figure 5.6: Top: T-slice at $s = -1.4244$. Bottom: S-slice at $t = 1.0446$. Structure is becoming visible in terms of clusters of similar data points.

of a 3D projection of the data set. As we change the t and s values we get a 3D grand tour. However these three values can be used in other ways. One suggestion is just to use two of these values as the coordinates of a 2D projection and the other one as a parameter to determine the size of the points used to represent the observations. This is, in some sense, also a 3D grand tour. Watching this grand tour we can also identify regularities in the data, this time as a consequence of the proximity of the points and the similarity in their size.

But now the idea is to combine whatever of the previous grand tours with another one. Once the points are positioned in a 3D projection, or in a 2D projection (with the adequate size), we can use the values obtained from another group of surfaces to change other parameters of the representation. The obvious choice, is to use *three* values as the colour components of the points. After a normalization step, we can get three values between 0 and 255 that we can use as the three values of a RGB colouring scheme. The points that are close according to the new grand tour will have similar colours. With this method we can save part of the work of brushing since the points and curves can be *auto-coloured* using the values obtained from the other grand tour. This use of our surfaces is in some way related to the image grand tour [178, 158], but instead of using multi-spectral images we use an arbitrary data set.

The user now can change independently the t and s values of one grand tour to get the positions of the points, or can change the values of the grand tour used to get the colour components, or can choose to change both simultaneously. That is, we can see an animation of the projection of the points moving in space, or we can see a static image of points changing their colours, or we can see both changes at the same time. This last method is the one used to obtain the grand tours whose snapshots appear in Figures 5.7 to 5.9. In Figure 5.9 we can see a snapshot of a grand tour we can call multidimensional, since we can see simultaneously the projection for all the t values as we change the s value. As a second grand tour we have used the principal component projection of the original data set. An animation of this and others grand tours of the some data set can be obtained in the web page `http://cis.paisley.ac.uk/fyfe-ci0/cgo/VIIP2004/`.

5.3 Extending Our Extension of Andrews' Curves

Our initial motivation to extend the Wegman and Solka curves was to jump from a 2D representation to a 3D one. Indeed in the previous section, we actually used

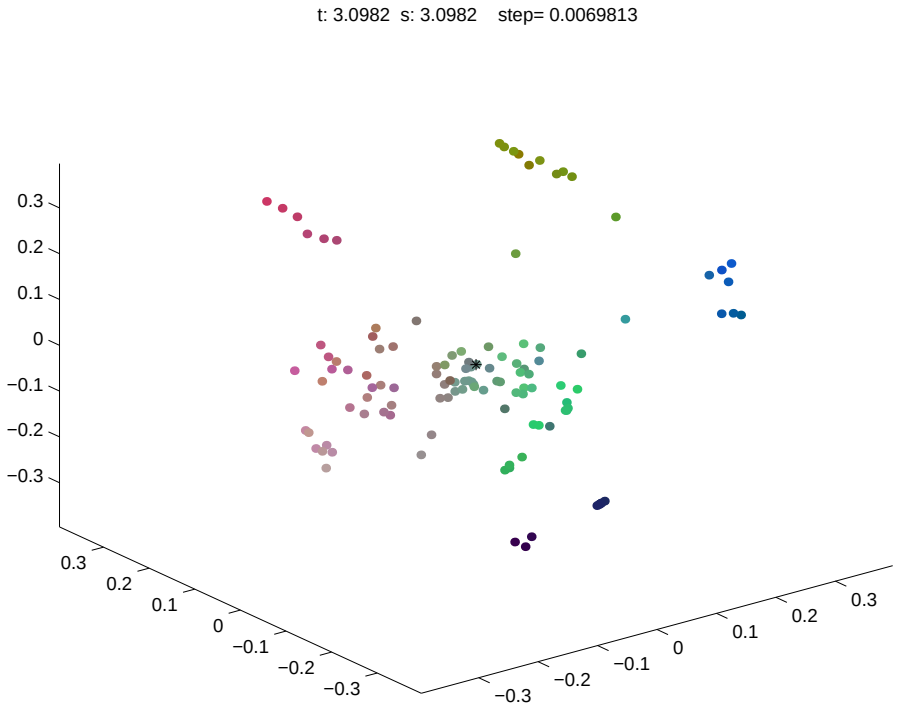


Figure 5.7: A snapshot of a 3D grand tour combined with the auto-colouring of the points (the full animation can be downloaded from `http://cis.paisley.ac.uk/fyfe-ci0/cgo/VIIP2004/GT3D.avi`). The position of the points is obtained from the surfaces constructed using the raw data, the colour comes from the surfaces constructed using the principal component projection of the data.

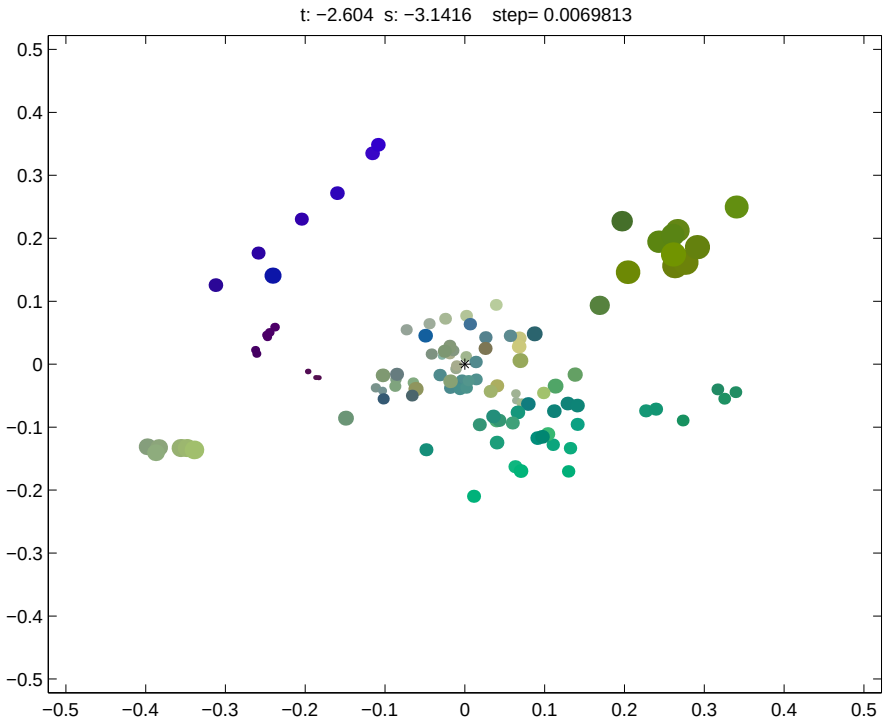


Figure 5.8: A snapshot of a 2D/size grand tour combined with the auto-colouring of the points (the full animation can be downloaded from `http://cis.paisley.ac.uk/fyfe-ci0/cgo/VIIP2004/GT2D.avi`). The position and size of the points come from the raw data, the colour from the principal component projection of the data.

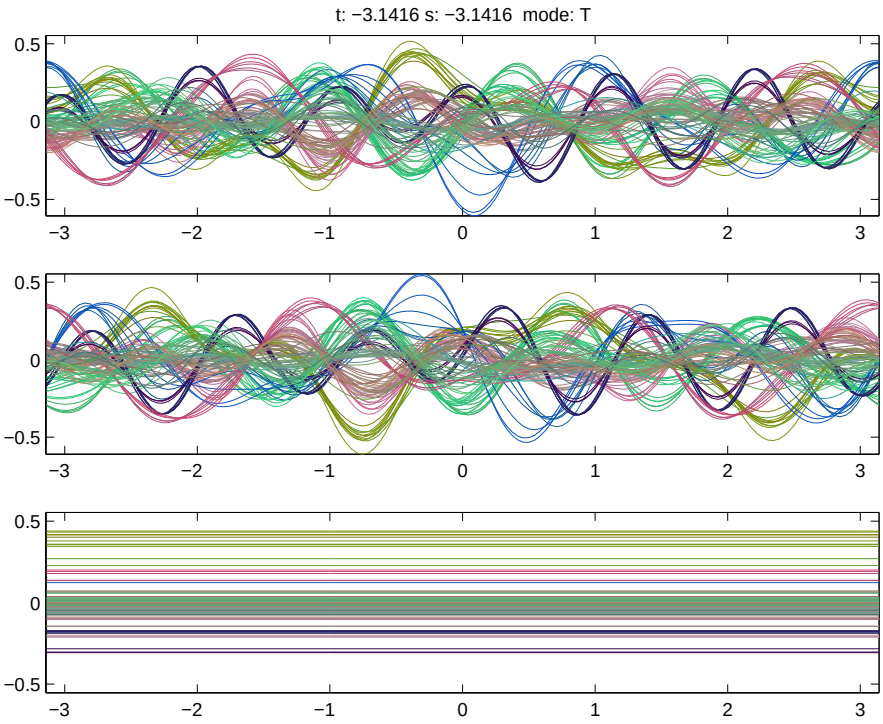


Figure 5.9: A snapshot of a n-D grand tour combined with the *auto-colouring* of the curves obtained from the principal components (the full animation can be downloaded from `http://cis.paisley.ac.uk/fyfe-ci0/cgo/VIIP2004/GTnD.avi`). This can save us having to brush some of the clusters.

a higher dimensional representation but took the dimensions greater than 3 from the principal component projections of the data. Thus we may be implicitly using the same information twice. This begs the question as to whether it is possible to create directly higher dimensional representations of the data set directly. We show below the obvious generalization to a four dimensional representation: each of the basis vectors shown is of length one and each is orthogonal to the other three i.e. we have an orthonormal basis. Now the fourth projection of the data can determine the colour, size or whatever feature we wish to use for the fourth characteristic.

$$
\begin{array}{llllll}
(& \cos(\lambda_1 t)\cos(\mu_1 s)\cos(\nu_1 r), & \cos(\lambda_1 t)\cos(\mu_1 s)\sin(\nu_1 r), & \cos(\lambda_1 t)\sin(\mu_1 s), & \sin(\lambda_1 t), & ...) \\
(& \sin(\lambda_1 t)\cos(\mu_1 s)\cos(\nu_1 r), & \sin(\lambda_1 t)\cos(\mu_1 s)\sin(\nu_1 r), & \sin(\lambda_1 t)\sin(\mu_1 s), & -\cos(\lambda_1 t), & ...) \\
(& \sin(\mu_1 s)\cos(\nu_1 r), & \sin(\mu_1 s)\sin(\nu_1 r), & -\cos(\mu_1 s), & 0 & , ...) \\
(& \sin(\nu_1 r), & -\cos(\nu_1 r), & 0 & 0 & , ...)
\end{array}
$$

We have arranged these four vectors in a manner which should make it clear that the proposal can be extended to any required dimensionality. This statement, though, must come with the caveat that the limit to the assistance which such a sequence of projections can make is liable to be determined by the difficulty which the human mind has on keeping track of several characteristics at one time. Even for five dimensions, one must keep track of position, speed, acceleration, colour and size simultaneously. Also we note that the final basis vector in any such basis is dependent on only two out of every five fields in the sample vector and so care must be taken in determining the order of the basis vectors.

5.4 Conclusions

In this chapter:

1. We have shown that Andrews' curves can be used for visualization of the results of data analysis by the SOM.

2. We have shown that we can perform a gross clustering of a data set using the SOM and then by selecting the data point for which a particular output neuron is the best matching unit, can iteratively find sub-clusters which the original SOM was unable to find.

3. We have shown that we can use the SOM as a pre-processing tool for Andrews' curves which originally were only of use on a few tens of data points at any one time. Now we suggest taking a large data set, using the SOM to find gross structure in the data and then using the Andrews' curves to find finer structure on subsets of the data at any one time.

4. We have discussed a new data mining technique so that, by using the interaction between the SOM, Andrews' curves and human operators, we can walk through a data set identifying local structures in the data set.

5. One of the major drawbacks of Andrews' curves is that, although they preserve the distance and the mean, their shapes depend on the order of the coefficients. Embrechts and Herzberg [42] suggest trying different orders for the coefficients. The combination proposed in this chapter can speed up exploratory data analysis. We can combine in the same device the information that comes from different ordinations or, as shown in Figures 5.7 to 5.9, the information that come from a particular order of the raw data and from the principal component projection of the data.

6. We have suggested an extension of our surfaces that give us a fourth parameter to add to the representations.

Chapter 6

Visualization in High Dimensional Feature Spaces

The use of kernel methods has been derived from the work of Vapnik [168], Burges [19] etc in the field of Support Vector Machines. Support Vector Machines for regression for example, perform a nonlinear mapping of the data set into some high dimensional (perhaps infinite dimensional) feature space in which we may then perform linear operations. Since the original mapping was nonlinear, any linear operation in this feature space corresponds to a nonlinear operation in data space.

Given the high dimensionality of the feature space, it is often difficult or even impossible to gain any geometric insight into the nature of the space. Interest has grown recently in the use of Principal Component Analysis in feature space [155, 146, 147, 144, 136, 148, 121, 145] but since the mapping from data space to feature space is uni-directional, it is often very difficult to gain an insight into the geometry of even the first few principal components except on toy problems, which typically use data of two dimensions.

In this chapter, we first review recent work on Kernel Principal Component Analysis (KPCA) [155, 156, 146, 147, 144, 136, 148, 121, 145] which has been the most frequently reported linear operation involving unsupervised learning in feature space.

Then we investigate the use of our surfaces as a way to get a picture of the structure of the feature space defined by a kernel matrix. We also use a combination of the surfaces and kernel methods to extend the kind of cluster that we can identify using the projections obtained from the Andrews' curves.

We finish the chapter presenting a method that, by means of bootstrapping, recovers the sparsity of Support Vector Machines in the context of Kernel Principal Component Analysis, reconciling the two extremes existing in the literature with respect to Kernel Principal Component Analysis: at one end, we have standard Kernel Principal Component Analysis which uses all the data points at the expense of losing the sparsity which we find in Support Vector Machines; at the other end, we have Sparse Kernel Principal Component Analysis which uses a single data point to represent the "Kernel Principal Component" direction.

6.1 Kernel PCA

This section is based very much on the analysis in [146, 147]. A very good Matlab simulation of Kernel PCA can be found at `http://www.kernel-machines.org/`.

In the next section, we show that simple Principal Component Analysis (PCA) may be performed on the samples of a data set in a particular way which will be useful in the performance of PCA in the nonlinear feature space.

6.1.1 The Linear Kernel

In Chapter 2, we presented the intuitive idea of PCA, that is finding of the directions of maximal variance. From a more formal point of view, PCA finds the eigenvectors and corresponding eigenvalues of the covariance matrix of a data set. Let $\chi = \{\mathbf{x}_1, ..., \mathbf{x}_M\}$ be iid (independent, identically distributed) samples drawn from a data source (that has been centered). If each $\mathbf{x}_i$ is n-dimensional, $\exists$ at most n eigenvalues/eigenvectors. Let C be the covariance matrix of the data set; then C is $n \times n$. Then the eigenvectors, $\mathbf{e}_i$, are n-dimensional vectors which are found by solving

$$C\mathbf{e}_i = \lambda_i \mathbf{e}_i \tag{6.1}$$

where λ_i is the eigenvalue corresponding to $\mathbf{e}_i$. We will assume the eigenvalues and eigenvectors are arranged in non-decreasing order of eigenvalues and each eigenvector is of length 1. We will use the sample covariance matrix as though it was the true covariance matrix and so

$$C \approx \frac{1}{M} \sum_{j=1}^{M} \mathbf{x}_j \mathbf{x}_j^T \tag{6.2}$$

Now each eigenvector lies in the span of χ; i.e. the set $\chi = \{\mathbf{x}_1, ..., \mathbf{x}_M\}$ forms a basis set (normally overcomplete since $M > n$) for the eigenvectors. So each $\mathbf{e}_i$ can be expressed as

$$\mathbf{e}_i = \sum_j \alpha_j^i \mathbf{x}_j \tag{6.3}$$

If we wish to find the principal components of a new data point $\mathbf{x}$ we project it on the eigenvectors previously found: the first principal component is $(\mathbf{x} \cdot \mathbf{e}_1)$, the second is $(\mathbf{x} \cdot \mathbf{e}_2)$, etc. These are the coordinates of $\mathbf{x}$ in the eigenvector basis. There are only n eigenvectors (at most) and so there can only be n coordinates in the new system: we have merely rotated the data set.

Now consider projecting one of the data points from χ on the eigenvector $\mathbf{e}_1$; then

$$\mathbf{x}_k \cdot \mathbf{e}_1 = \mathbf{x}_k \cdot \sum_j \alpha_j^1 \mathbf{x}_j = \sum_j \alpha_j^1 \mathbf{x}_k \cdot \mathbf{x}_j \tag{6.4}$$

Now let K be the matrix of dot products. Then $K_{ij} = \mathbf{x}_i \cdot \mathbf{x}_j$.

Multiplying both sides of (6.1) by $\mathbf{x}_k$ we get

$$\mathbf{x}_k \cdot (C\mathbf{e}_1) = \lambda_1 (\mathbf{e}_1 \cdot \mathbf{x}_k) \tag{6.5}$$

and using the expansion for $\mathbf{e}_1$, and the definition of the sample covariance matrix, C, gives

$$\mathbf{x}_k \cdot \left(\frac{1}{M} \left(\sum_{j=1}^{M} \mathbf{x}_j \mathbf{x}_j^T \right) \left(\sum_t \alpha_t^1 \mathbf{x}_t \right) \right) = \lambda_1 \left(\left(\sum_t \alpha_t^1 \mathbf{x}_t \right) \cdot \mathbf{x}_k \right)$$

$$\frac{1}{M} \left(\sum_{j=1}^{M} \mathbf{x}_k \cdot \mathbf{x}_j \mathbf{x}_j^T \right) \left(\sum_t \alpha_t^1 \mathbf{x}_t \right) = \lambda_1 \left(\sum_t \alpha_t^1 \mathbf{x}_t \cdot \mathbf{x}_k \right)$$

$$\frac{1}{M} \sum_{j=1}^{M} \left(\mathbf{x}_k \cdot \mathbf{x}_j \sum_t \alpha_t^1 \mathbf{x}_j^T \mathbf{x}_t \right) = \lambda_1 \left(\sum_t \alpha_t^1 \mathbf{x}_t \cdot \mathbf{x}_k \right)$$

Instead of the dot product we can use the elements of the matrix K

$$\frac{1}{M} \sum_{j=1}^{M} \left(K_{kj} \sum_t \alpha_t^1 K_{jt} \right) = \lambda_1 \sum_t \alpha_t^1 K_{tk}$$

Since the above relation holds for all k, finally we get

$$\frac{1}{M} K^2 \alpha^1 = \lambda_1 K \alpha^1 \tag{6.6}$$

Now it may be shown [147] that all interesting solutions of this equation are also solutions of

$$K\alpha^1 = M\lambda_1\alpha^1 \tag{6.7}$$

whose solution is that α^1 is the principal eigenvector of K. And we can obtain $\mathbf{e}_1$ using equation 6.3. So we have two methods of obtaining the eigenvectors, directly using the covariance matrix, or indirectly using the matrix of dot products. In a normal context there is no advantages of using the second methods, but let us see what happen if we preprocess the data using a non-linear transformation.

6.1.2 Non Linear Kernels

Now we preprocess the data using $\Phi : \chi \rightarrow \mathcal{F}$. So $\mathcal{F}$ is now the space spanned by $\Phi(\mathbf{x}_1), ..., \Phi(\mathbf{x}_M)$. The above arguments all hold and the eigenvectors of the covariance matrix can be found from the eigenvectors of the dot product matrix $K_{ij} = (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j))$. But now the *Kernel Trick*: provided we can calculate K we don't need the individual terms $\Phi(\mathbf{x}_i)$.

One issue that we must address in feature space is that the eigenvectors should be of unit length. Let $\mathbf{v}_i$ be an eigenvector of C. Then $\mathbf{v}_i$ is a vector in the space $\mathcal{F}$ spanned by $\Phi(\mathbf{x}_1), ..., \Phi(\mathbf{x}_M)$ and so can be expressed in terms of this basis. This is an at most M-dimensional subspace of a possibly infinite dimensional space which gives computational tractability to the kernel algorithms. Then

$$\mathbf{v}_i = \sum_{j=1}^{M} \alpha_j^i \Phi(\mathbf{x}_j) \tag{6.8}$$

for eigenvectors $\mathbf{v}_i$ corresponding to non-zero eigenvalues. Therefore

$$\begin{aligned}
\mathbf{v}_i^T\mathbf{v}_i &= \sum_{j,k=1}^{M} \alpha_j^i \Phi(\mathbf{x}_j)^T \Phi(\mathbf{x}_k)\alpha_k^i \\
&= \sum_{j,k=1}^{M} \alpha_j^i K_{jk}\alpha_k^i \\
&= \alpha^i \cdot (K\alpha^i) \\
&= M\lambda_i\alpha^i \cdot \alpha^i
\end{aligned}$$

Now α^i are (by definition of the eigenvectors of K) of unit magnitude. Therefore since we require the eigenvectors to be normalized in feature space, $\mathcal{F}$, i.e. $\mathbf{v}_i^T\mathbf{v}_i =$

1, we must normalize the eigenvectors of K, α^i, by dividing each by $\sqrt{M\lambda_i}$.

Now we can simply perform a principal component projection of any new point $\mathbf{x}$ by finding its projection onto the principal components of the feature space, $\mathcal{F}$. Thus

$$\mathbf{v}_i \cdot \Phi(\mathbf{x}) = \sum_{j=1}^{M} \alpha_j^i \Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}) = \sum_{j=1}^{M} \alpha_j^i K(\mathbf{x}_j, \mathbf{x}) \tag{6.9}$$

And the above argument shows that any operation which can be defined in terms of dot products can be Kernelised. We will in subsequent sections use the same kernel trick to perform projection that can reveal the structure of the data.

6.2 Using Andrews' Curves to Project the Kernel Space

The high dimensionality of the feature space makes it difficult to have an idea about the structure of that space. In this section we use the kernel trick to obtain a projection of the points in the feature space. Then we use this projection to facilitate the identification of certain kinds of clusters.

6.2.1 The Basic Idea

The high dimensional space is the feature space defined by a mapping Φ, and we will project the points onto a vector $\mathbf{w}_1$:

$$y_1 = \mathbf{w}_1^T \Phi(\mathbf{x})$$

This vector onto we are going to project the data can be expressed in terms of the other points in feature space; that is $\mathbf{w}_1$ can be expressed as linear combination of other points in the feature space: $\mathbf{w}_1 = \alpha_1^1 \Phi(\mathbf{x}_1) + \alpha_2^1 \Phi(\mathbf{x}_2) + \ldots \alpha_N^1 \Phi(\mathbf{x}_N)$, so we have:

$$y_1 = \left(\sum_{i=1}^{N} \alpha_i^1 \Phi(\mathbf{x}_i) \right)^T \Phi(\mathbf{x}) = \left(\sum_{i=1}^{N} \alpha_i^1 \Phi^T(\mathbf{x}_i) \right) \Phi(\mathbf{x})$$

Now we can define this dot product in feature space in terms of the Kernel matrix, K:

$$y_1 = \left(\sum_{i=1}^{N} \alpha_i^1 \Phi^T(\mathbf{x}_i) \Phi(\mathbf{x}) \right) = \left(\sum_{i=1}^{N} \alpha_i^1 K(\mathbf{x}_i, \mathbf{x}) \right)$$

In order to manipulate the actual $\mathbf{w}_1$, which is potentially infinite dimensional, we are going to change α^1 using the same technique (Andrews' curves and extensions thereof) which we have previously used in data space to identify structure. That is, α^1 will be a vector like $\big(\sin(t), \cos(t), \sin(2t), \cos(2t), \ldots\big)$ or some of the variants we have previously discussed. We may typically decide that a two or three dimensional projection of the data is required in which case we will augment α^1 with α^2 or α^3 to get a three dimensional projection, (y_1, y_2, y_3), of this high (infinite) dimensional space.

6.2.2 A First Example

To illustrate the method, we begin with a first artificial data set composed of five clusters of points. The points originally were generated as noisy clusters around five centers in a 2 dimensional space; we embed this in a ten dimensional space where each of the other 8 dimensions simply consisted of Gaussian noise; subsequently all the data points were rotated using a randomly generated rotation matrix.

Thus the natural basis in which the points are presented exhibits no clear clustering but the optimal space for viewing the clusters (the original two dimensional space in which they were created) can be found by a simple rotation of the 10 dimensional space.

We illustrate the method using the linear kernel: we see in Figure 6.1, the Extended Andrews Curves for this data set. We have a three degrees of freedom mapping which is one degree of freedom more than is necessary for this simple problem; in the left diagram, we see that the five clusters have been clearly identified while on the right, we show how we may move along the s or t parameterized spaces to find the optimal values.

We also show in Figure 6.2 the equivalent diagram for a nonlinear kernel (actually the radial kernel which we will use to some effect later). From both parts of that diagram, it is apparent that some differentiation into the different clusters is taking place but the result is much less clear than in the linear case. This suggests that, as an exploratory data tool, linear mappings should be used first and all possible structure identified from these before we advance to nonlinear mappings.

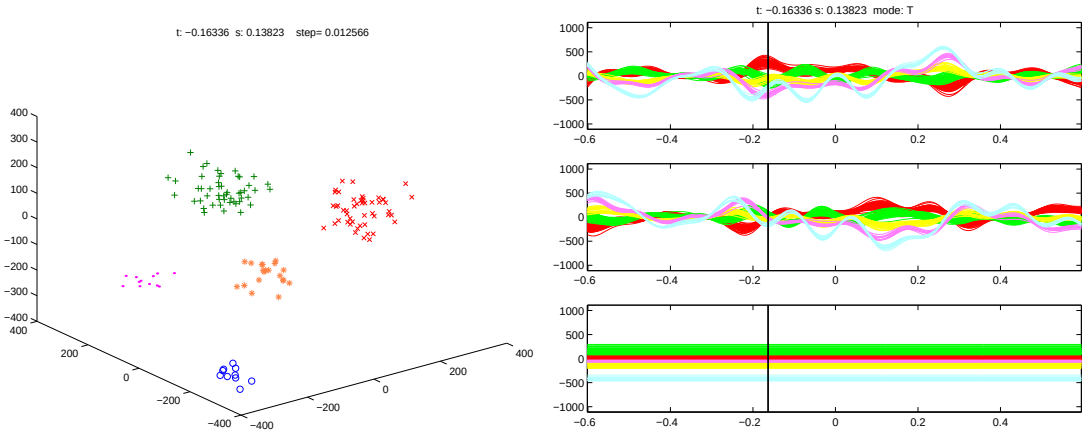


Figure 6.1: Left: a projection of the points of the five clusters using a linear kernel. Right: the T-slice of our surface, at $s = 0.13823$, corresponding to the projection of the figure on the left.

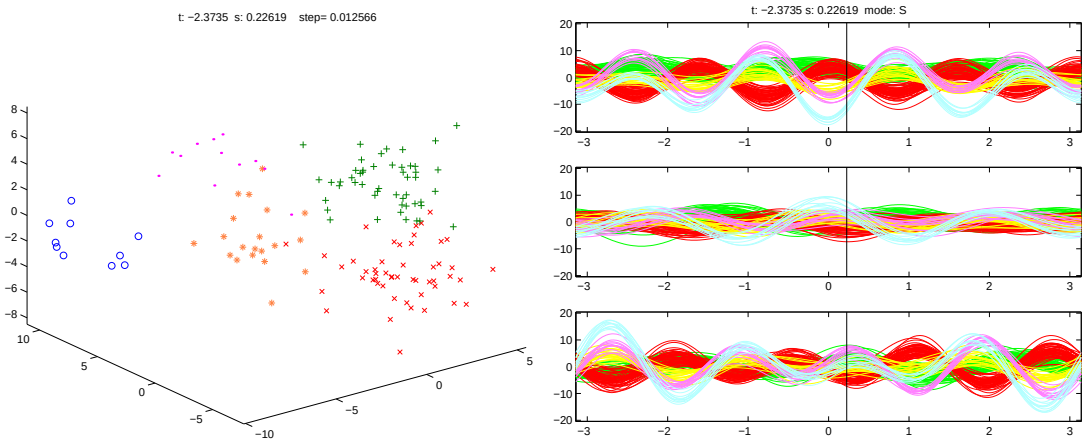


Figure 6.2: Left: a projection of the points of the five clusters. Right: the S-slice of our surface, at $t = -2.3735$, corresponding to the projection of the figure on the left.

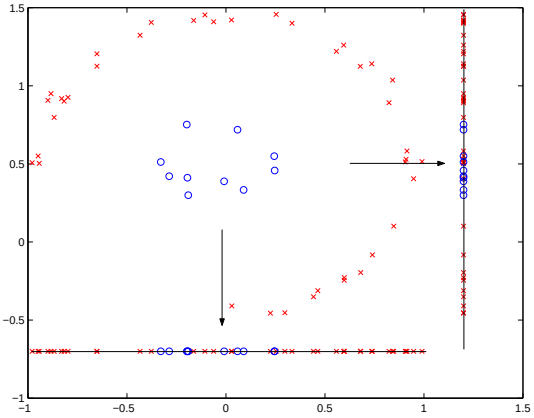


Figure 6.3: Two one-dimensional projections of a two-dimensional data set.

6.2.3 The Need for Non Linearities

We can easily create data sets for which no linear projection can be used to differentiate clusters. Consider the extreme case of Figure 6.3. Now we have two dimensional data and two clusters, one of which lies completely within the other. We have illustrated the difficulties which this data set presents by showing two one dimensional projections; it should be clear that no one dimensional linear projection exists which will differentiate between these two clusters. Clearly in this simple example we can see the structure in a two dimensional projection but if we increase the dimensionality of the data set, this no longer becomes an option.

Figure 6.4 shows the difficulty which our standard Andrews' curves method has with this data set. We have illustrated in the right diagram how the central set of points changes as a group as we move along the Extended Andrews' curves but this does not help us in an exploratory data analysis in which we do not *a priori* have information on which data points belong to which cluster; we see that the central set of data points is effectively occluded by the other data points.

However we can project the data into a higher dimensional feature space and search for structure in that space. In Figures 6.5 and 6.6, we show the result of using the Extended Andrews Surfaces to project the same artificial data set as in Figure 6.4. In this case, we have used three orthogonal vectors to get the projection, so that by changing the values of t and s we can obtain a three dimensional grand tour. We can see how easy it is now to identify the cluster. The different behavior of the curves for the points in the different clusters is evident

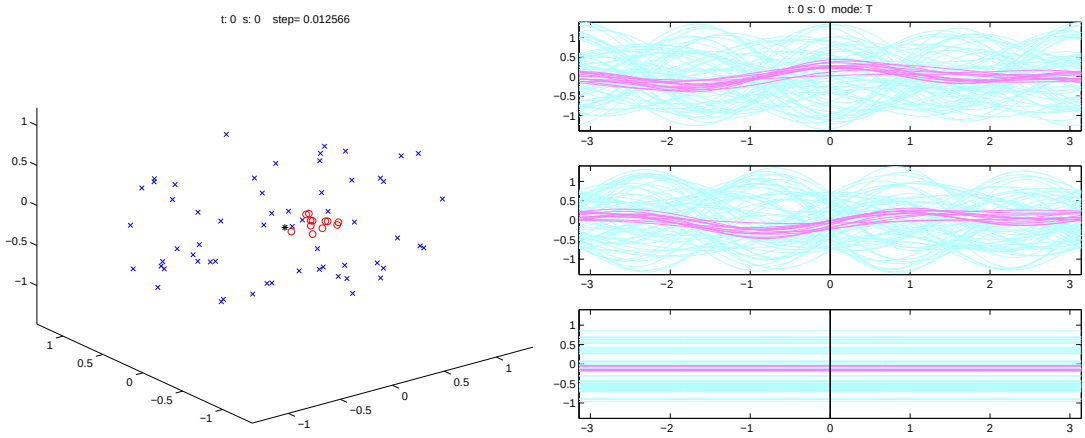


Figure 6.4: Left: The projection of a data set with two clusters that are difficult to identify (the circle points correspond to points within the internal hypersphere, the crosses are those from the surface of the hypersphere). Right: the T-slice of our surface, at $s = 0$, corresponding to the projection of figure on the left (the darker curves correspond to points within the internal hypersphere).

in the t-slices. Figure 6.5 was created using a radial basis kernel:

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{-\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2}\right) \tag{6.10}$$

and Figure 6.6 was created using a polynomial kernel:

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^d \tag{6.11}$$

6.3 Applying Bagging to Obtain Sparse Kernel Principal Components

Figure 6.7 shows the clustering ability of Kernel PCA with the Gaussian Kernel in equation 6.10. The data set comprises 3 Gaussian clouds each of 30 points. The figure shows the contours of equal projection onto the first 8 KPCA directions. Note that linear PCA would only be able to extract 2 principal components; however because the kernel operation has moved us into a high dimensional space in a nonlinear manner, there may be up to 90 non-zero eigenvalues. The three clusters can be clearly identified by projecting the data points onto the first two eigenvectors. Subsequent Kernel Principal Components split the clusters into

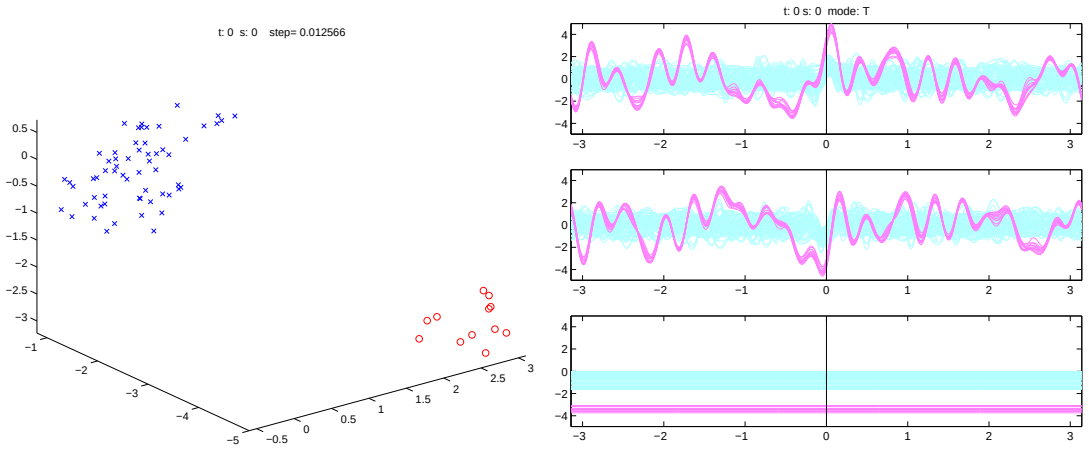


Figure 6.5: Left: a projection of the same data set of Figure 6.4 after transforming using a radial basis kernel. Right: the t-slice of the surface, at $s = 0$, corresponding to the projection of the figure on the left. (the darker curves correspond to points within the internal hypersphere).

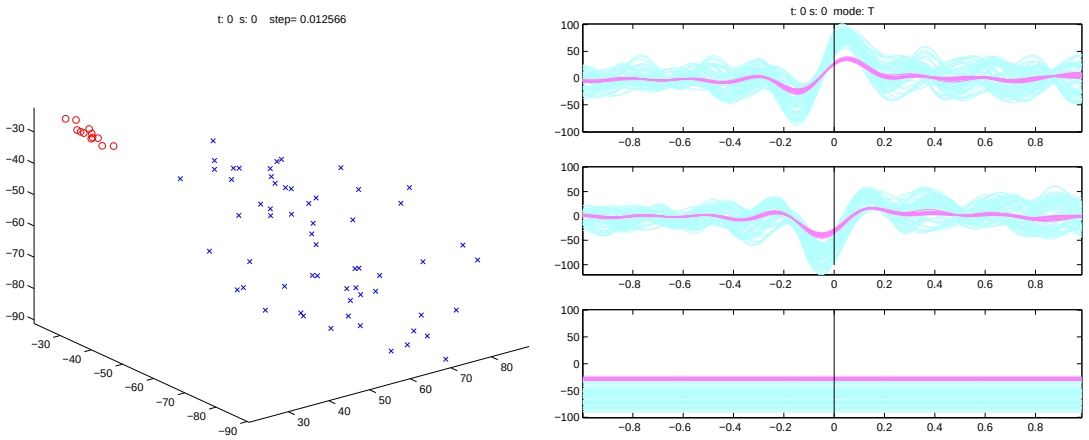


Figure 6.6: Left: a projection of the same data set of Figure 6.4 after transforming using a polynomial kernel. Right: the t-slice of the surface, at $s = 0$, corresponding to the projection of the figure on the left (the darker curves correspond to points within the internal hypersphere).

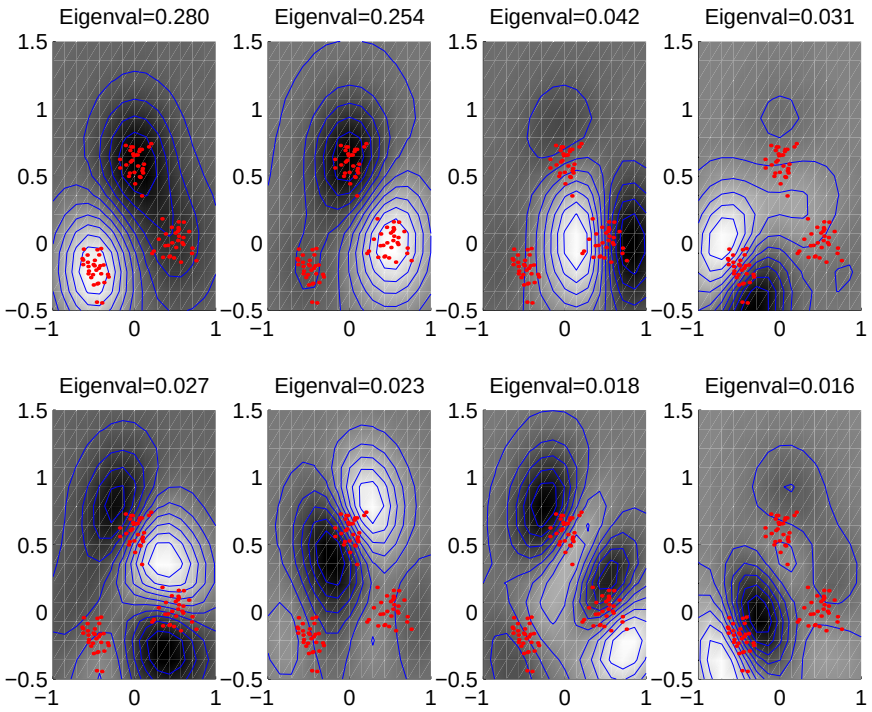


Figure 6.7: The 3 clusters data set is shown as individual points. The contours are contours of equal projection on the respective Principal Components. The first two principal components are sufficient to differentiate between the three clusters; the others slice the clusters internally and have much less variance associated with them.

sections.

However Figure 6.8 shows the components of the eigenvectors in feature space. We see why the first two projections were so successful at identifying the three clusters but we note that there is a drawback to the method if we were to use this method to identify cases: each eigenvector is constructed with support from projections of very many points. What we really wish is to identify individual points in terms of their importance.

6.3.1 Sparse Kernel Principal Component Analysis

It has been suggested Smola et al. [155] that we may reformulate the Kernel PCA problem as follows: let the set of permissible weight vectors be

$$\mathbf{V} = \left\{ \mathbf{w} : \mathbf{w} = \sum_{i=1}^{M} \alpha^i \Phi(\mathbf{x}_i), \text{ with } \sum_i ||\alpha^i||^2 \leq 1 \right\} \tag{6.12}$$

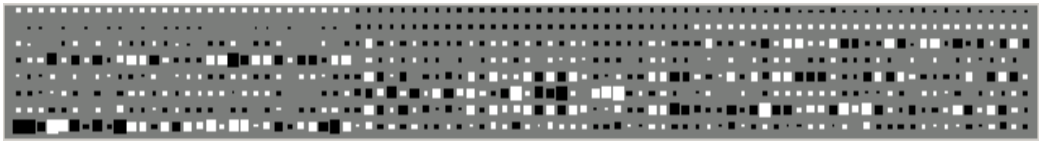


Figure 6.8: The first eight eigenvectors found by Kernel PCA. Each eigenvector has elements from every data point.

Then the first principal component is

$$\mathbf{v}_1 = \arg\max_{\mathbf{v}\in\mathbf{V}} \frac{1}{M}\sum_{i=1}^{M} |\mathbf{v}.\Phi(\mathbf{x}_i)|^2 \tag{6.13}$$

for centered data. This is the basic KPCA definition which we have used above. Now we may ask whether other sets of permissible vectors may also be found to be useful. Consider[1]

$$\mathbf{V}_{LP} = \left\{ \mathbf{w} : \mathbf{w} = \sum_{i=1}^{M} \alpha^i \Phi(\mathbf{x}_i), \text{ with } \sum_i |\alpha^i| \leq 1 \right\} \tag{6.14}$$

This is equivalent to a sparsity regularizer used in supervised learning and leads to a type of kernel feature analysis

$$\mathbf{v}_1 = \arg\max_{\mathbf{v}\in\mathbf{V}_{LP}} \frac{1}{M}\sum_{i=1}^{M} |\mathbf{v}.\Phi(\mathbf{x}_i)|^2 \tag{6.15}$$

Now Smola et al. [155] show that the solutions of equation (6.15) are to be found at the corners of the hypercube determined by the basis vectors, $\Phi(\mathbf{x}_i)$. Therefore all we require to do is find that element

$$\mathbf{x}_k = \arg\max_{\mathbf{x}_t\in\chi} \sum_{i=1}^{M} |\Phi(\mathbf{x}_k).\Phi(\mathbf{x}_i)|^2 = \arg\max_{\mathbf{x}_t\in\chi} \sum_{i=1}^{M} |K_{ki}|^2 \tag{6.16}$$

[1]Smola et al. [155] point out that this system may be generalized by considering the l_p norm to create permissible spaces

$$\mathbf{V}_P = \left\{ \mathbf{w} : \mathbf{w} = \sum_{i=1}^{M} \alpha^i \Phi(\mathbf{x}_i), \text{ with } \sum_i ||\alpha^i||^p \leq 1 \right\}$$

for any $p \in (0, 1]$

which again requires us only to evaluate the kernel matrix. So that $\mathbf{v}_1$ is a vector all of zeros except for a one in the k^{th} position.

6.3.2 Bootstrapping and Bagging

Bootstrapping [41] is a simple and effective way of estimating a statistic of a data set. Let us suppose we have a data set, $D = \mathbf{x}_i, i = 1, \ldots, N$. The method consists of creating a number of pseudo data sets, D_i, by sampling from D with uniform probability with replacement of each sample. Thus each data point has a probability of $\left(\frac{N-1}{N}\right)^N \approx 0.368$ of not appearing in each bootstrap sample, D_i. It has been applied to predictors in general [16, 17] (and given the name bagging ("bootstrap aggregation")) and artificial neural networks in particular, [40] but almost always to the popular backpropagation algorithm.

Bagging then, consists of selecting B data sets, $D_1, \ldots, D_B$ from D by randomly selecting a member of D with replacement. Each data set will almost certainly contain only some members of the original data set D and the remaining members of D can be used to determine the optimal parameters of the predictor which is trained on the data set D_i. Furthermore, the elements of $T_i = D - D_i$ can be used to assess how accurate the training of the individual predictor is liable to be.

6.3.3 Application to Sparse Kernel Principal Component Analysis

The SKPCA method of [155] identifies a single point as the most important point in the data set for the identification of the principal component filter. Now we create 20 bags by sampling from the data set with replacement. We must be aware of two effects:

1. Sometimes the most important data point will not be in the bag.

2. Sometimes it will be, but will no longer be the most important since there may be a different point with a higher sum of scalar products with the points actually in the bag.

Thus, running the Sparse KPCA method on the twenty bags will find several different most important points i.e. will find different principal components. However some points will tend to be found more than once (they tend to have a

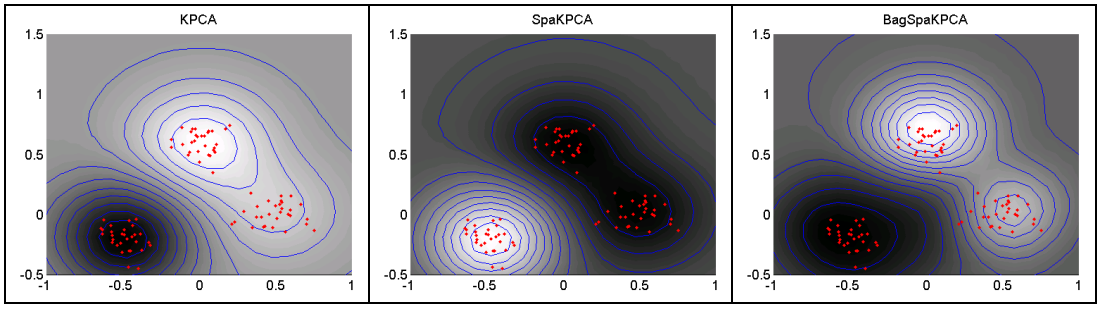


Figure 6.9: First principal component. Left: using the ordinary Kernel PCA. Center: using the Sparse Kernel PCA. Right: using the Bagged Sparse Kernel PCA.

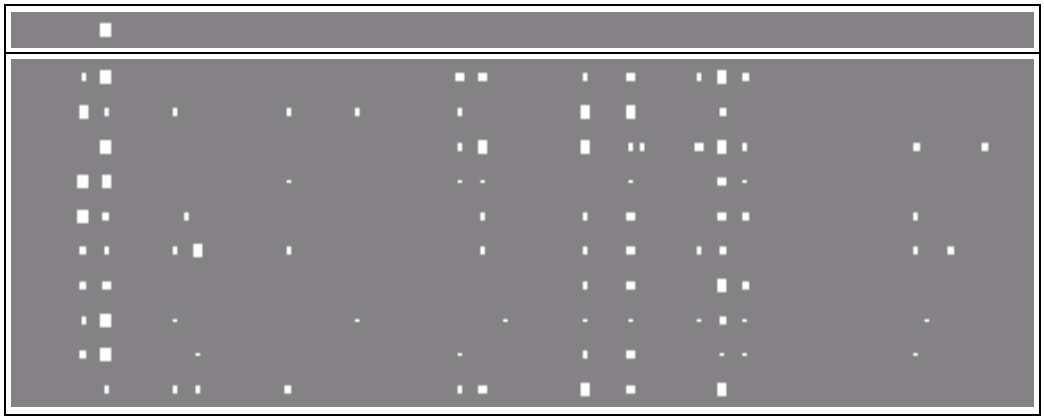


Figure 6.10: First eigenvectors found by Sparse Kernel PCA (first line) and the Bagged Sparse Kernel PCA (lines second to last).

high scalar product within the original data set) while others will never be used in this manner (they are, perhaps, at the edge of a cluster and never have a high scalar product with any other point). Now we weight the SKPCA estimate by the proportion of times each data point appears as the SKPCA. Results on the same data as before are shown in Figure 6.9. We see that the Bagged SKPCA is qualitatively similar to the KPCA. Also the first line of Figure 6.10 shows the SKPCA direction: it identifies one of the ninety data points as the most important. Underneath that, we show the results from 10 simulations, each of 20 bags. We see that we have a sparse representation; reassuringly, the single SKPCA data point features strongly in each bag and also we see that the bagged results tend to be similar. We thus have created a method for the identification of a few most important points which are determining the principal components in kernel space.

6.4 Conclusions

We have presented a way to try to visualize what is happening in the feature space eluding some difficulties that are created by the high dimensionality of that space.

To achieve that goal we have used the same trick as in kernel methods. We have used a kernel metric that lets us not to care about the real mapping that is taking place.

We have illustrated the method with two different artificial data sets. In one of them the identification of the clusters is very difficult without using the non linearities supplied by the kernel methods.

Finally, we have used the method of bagging for creating a sparse but not too sparse, representation which identifies a handful of data points as important and, in addition, weights their importance in terms of the number of times they are deemed to be most important.

Chapter 7

Conclusions

7.1 Summary and Remarks

The novel work in this thesis can be divided into three areas.

1. The first one has to do with the Exploratory Projection Pursuit (EPP) methods and appears in Chapter 3. There, we have performed a comparative study of two existing EPP neural networks, the Output Functions EPP algorithm and the Maximum Likelihood EPP algorithm. We have shown empirically that the first is slower to converge but gives us more accurate results while the second, on the other hand, does exhibit very fast convergence but with much less accuracy. We have developed a new EPP algorithm which is a combination of these two, and we have shown experimentally that it converges faster and gives better two-dimensional projections than the other two. In this comparison process, we have also developed a comparison framework for EPP algorithms that uses, in a novel manner, a set of indices previously used in the context of cluster validation. In this way, we have avoided the bias of using only few projections, and the subjective judgement of the person who compares the goodness of the projections.

   We have also extended the Maximum Likelihood method so that it can now identify skewed distributions though this met with only limited success: we have very fast convergence to approximately the correct direction but the convergence though stable is very approximate.

   Finally, we have initiated the transition to the next area, with a first utilization of Andrews' curves. We have used them to choose a promising pro-

jection which gives the initial weights for the EPP networks. This weights initialization improves the convergence of the algorithms, and even enables us to use algorithms that, without it, do not exhibit convergence. Besides, it has let us discover the inherent inaccuracy of the maximum Likelihood algorithm in that, even when the parameters are set to their optimal values, they will still diverge from these within the algorithm.

2. In Chapters 4 and 5 we have focused on Andrews' curves. These curves can be thought of as a mono dimensional grand tour (a pseudo grand tour, really). First we have reviewed one extension of these curves that lets us construct a bi-dimensional grand tour, and then we have extended it to the three dimensional case; then, we have compared both extensions. We have shown that most of the properties of the original Andrews' curves also hold for our extension. We have illustrated its uses with a real data set and, in combination with techniques such as brushing and linking, it enables us to identify most of the labelled clusters of the data set, and also to discover new clusters in the unlabelled samples, and identify sub-clusters in one labelled cluster.

   In Chapter 5 we go some steps further with our extension. We have dealt with two problems of Andrews' curves: the limitations in the size of the data set we can use with them, and their dependency on the order of the variables. To solve the former we have proposed using the SOM. The model vectors of a SOM training on a data set give us a convenient interface to choose with points we want to represent using the Andrews' curves. To solve the latter, Embrechts and Herzberg [42] suggest trying different orders for the coefficients. We have proposed a way of speeding up that process by means of combining in the same display the information that comes from different ordinations of variables (or that come from a particular order of the raw data and from the principal component projection of the data). Finally, we have given a generalization of our extension to Andrews' curves that provides us with a fourth projection of the data which we can use to determine others features of the display different from the space coordinates (for example, color or size).

3. In Chapter 6 we have moved into the feature space. First, we have investigated the use of Andrews' curves (and extensions therefore) as a mean of visualizing that space. We have also used bootstrapping and bagging

to create a method for the identification of a few most important points which are determining the principal components in kernel space. This give a sparse representation of the data set, but not as sparse as the Sparse Kernel Principal Component Analysis.

7.2 Limitations and Further Research

- Explanation of the behaviour of Maximum Likelihood algorithm.

  The initialization of EPP networks was performed using an artificial data set with known structure in one of the dimensions. The simplicity of the data set makes the analysis of the results clearer. Indeed that lets us discover that the Maximum Likelihood algorithm is responding to other statistics than it was designed for. However, a more exhaustive investigation, with more complicated data set (for example, with structure in more than one direction), could shed some light on the strange behaviour of the algorithm. There is also of course the need to understand its shortcomings analytically.

- Use of α-blending.

  The same technique used with parallel coordinates to show the density of lines, α-blending, could be implemented with Andrews's curves, so leading with high volume data sets . In Figure 7.1 we can see the result of applying α-blending to Andrews' curves. The utility of this technique in the context of Andrews' curves is something that is worth future research.

- Order of variables, change and watch display.

  To overcome the problem of the Andrews' curves with the strong dependence on the order of the variables it has been proposed [42] to try different orderings. The interactive change of the order of variables could be incorporated to our prototype program.

- New orthogonal curves based in other variations.

  With our surfaces we have extended the idea of Wegman and Shen who gives a vector perpendicular to the one originally proposed by Andrews. In Section 4.1 we give a derivative perspective of its formulation. The same idea could be used with the curves proposed by Khatree and Naik [101].

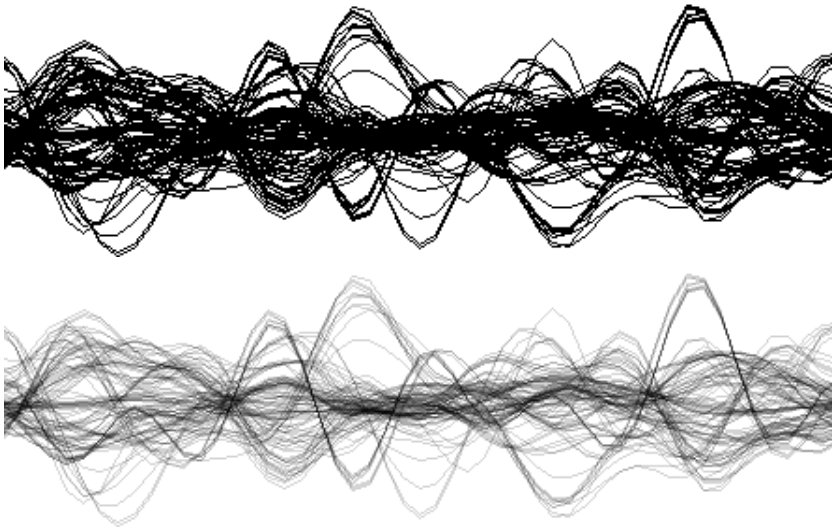


Figure 7.1: Andrews' curves without (upper) and with (lower) α-blending.

Now, the two orthogonal vectors that will give a bi-dimensional grand tour are:

$$\begin{aligned}\mathbf{w}_1 \propto \Big(&\big(\sin(\lambda_1 t)+\cos(\lambda_1 t)\big),\big(\sin(\lambda_1 t)-\cos(\lambda_1 t)\big),\\ &\big(\sin(\lambda_2 t)+\cos(\lambda_2 t)\big),\big(\sin(\lambda_2 t)-\cos(\lambda_2 t)\big),\cdots\Big)\\ \mathbf{w}_2 = \frac{\partial \mathbf{w}_1}{\partial t} \propto \Big(&\lambda_1\big(-\cos(\lambda_1 t)+\sin(\lambda_1 t)\big),\lambda_1\big(-\cos(\lambda_1 t)-\sin(\lambda_1 t)\big),\\ &\lambda_2\big(-\cos(\lambda_2 t)+\sin(\lambda_2 t)\big),\lambda_2\big(-\cos(\lambda_2 t)-\sin(\lambda_2 t)\big),\cdots\Big)\end{aligned}$$

where we have used a slightly changed version of the vector used to obtain (2.28).

- Interactive calculation of confidence intervals.

  We have not given an in depth analysis of the variance preservation in our surfaces. First, as we noted in Section 2.4.2 that property is less important than the mean and distance preservation, since it is conditional on the lack of correlation of the variables. Second the analysis of that property is not so easy in our surfaces. Third, even obtaining an expression for the intervals, that expression is going to depend on the point we use to obtain a slice, and within that slice it will be different along the x axis (although we can imagine a pointer mechanism that shows this variable intervals along the selected curve, in that way we can confirm or discard the hypothesis of other

curves belonging to the same group as the selected one).

- Sparse Kernel PCA.

  In the Section 6.3 we have proposed a new way to obtain a sparse representation of the data. Future work in this issue could be the comparison of the projections of KPCA, SKPCA and the bagged SKPCA to evaluate what is being lost or gained by the respective projections.

Appendix A

Color Plates

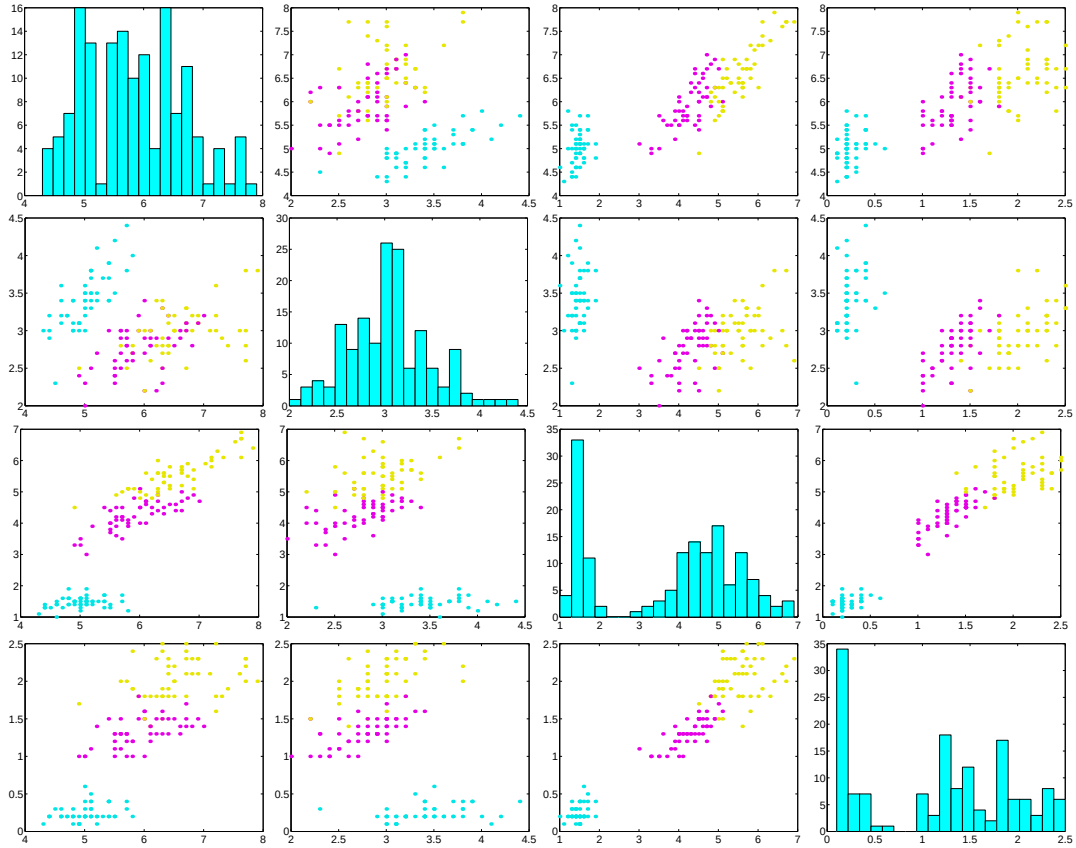


Figure 2.6: Scatterplot for the iris dataset

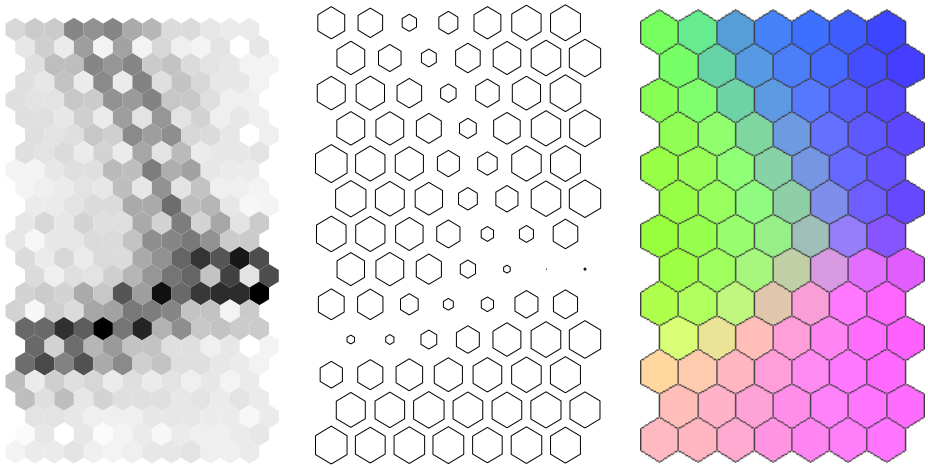


Figure 2.8: Examples of SOM visualizations methods.

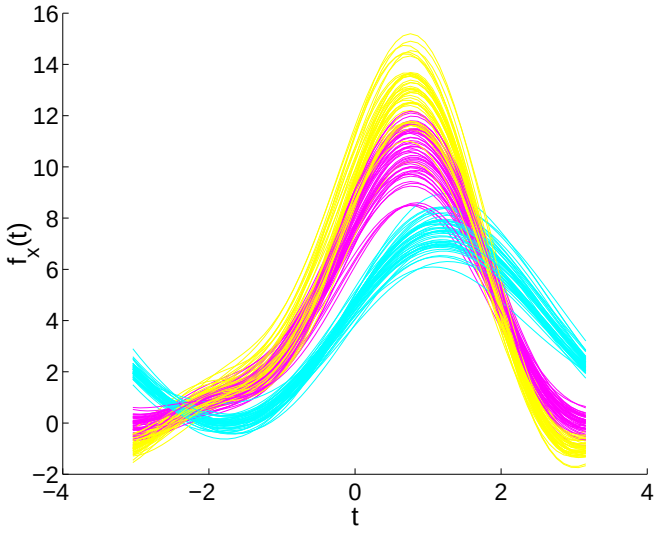


Figure 2.12: An Andrews' plot of the iris data set. It is clear that one type of iris is distinct from the other two but differentiating between the other two is less easy.

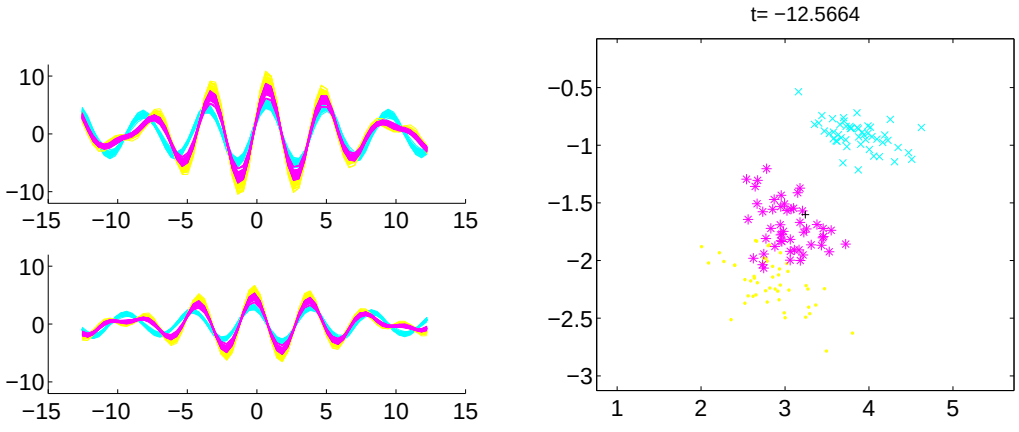


Figure 4.1: *Left*: the two Wegman's curves for the iris data set ($-4\pi \leq t \leq 4\pi$). *Right*: a snapshot of the grand tour.

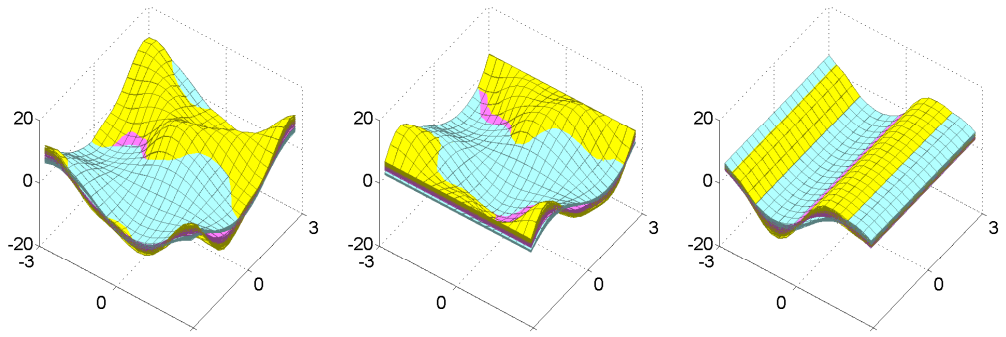


Figure 4.3: The three groups of surfaces for the Iris data set.

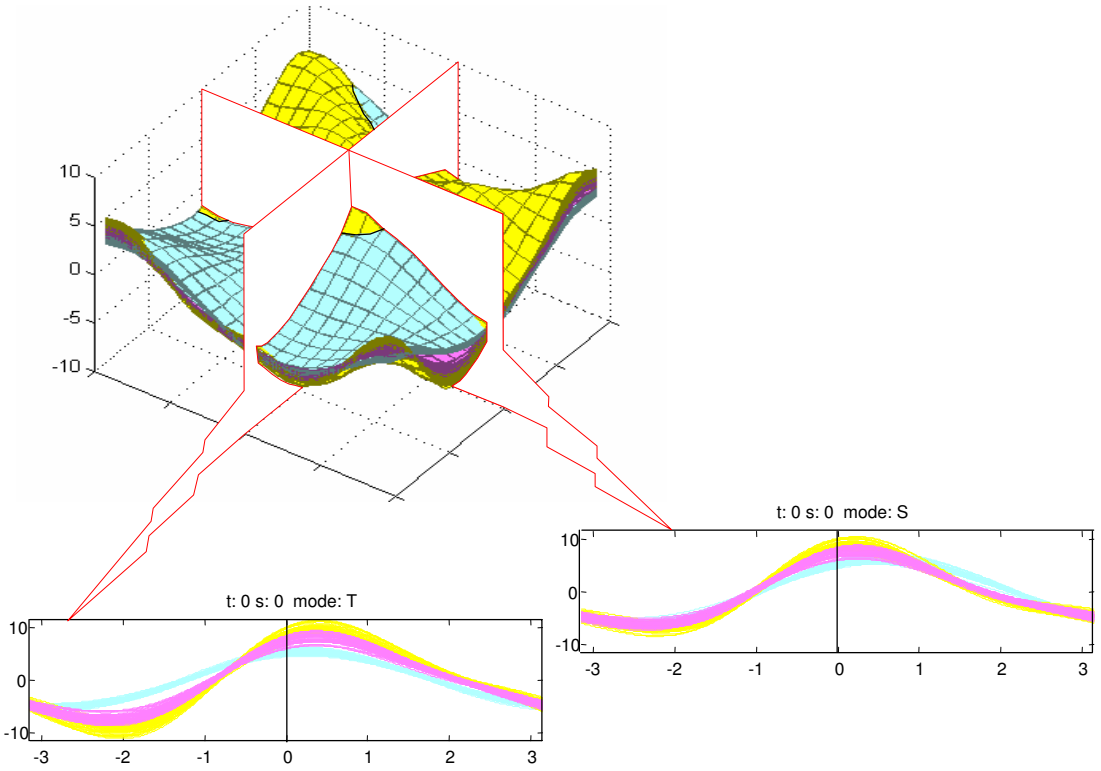


Figure 4.4: One of the surfaces and its T and S slices.

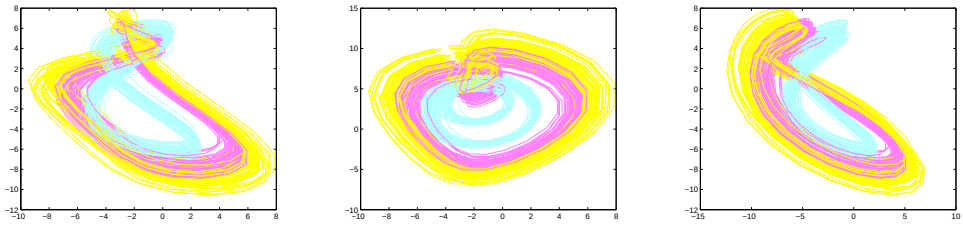


Figure 4.5: Three different perspectives (front, top and side views) of the curves obtained from the Iris data set when $t = s$.

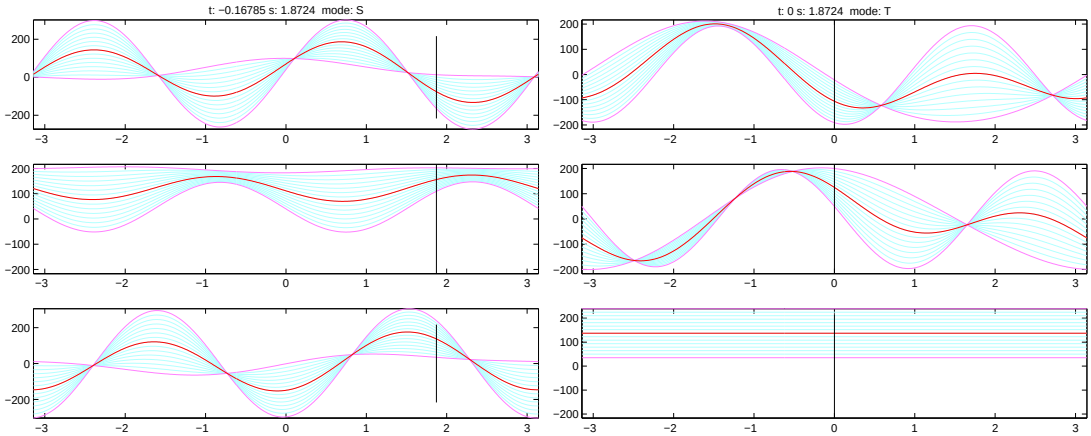


Figure 4.6: Surfaces for points (in cyan) equally spaces between two points (in pink). The surface associated to the mean appears in red.

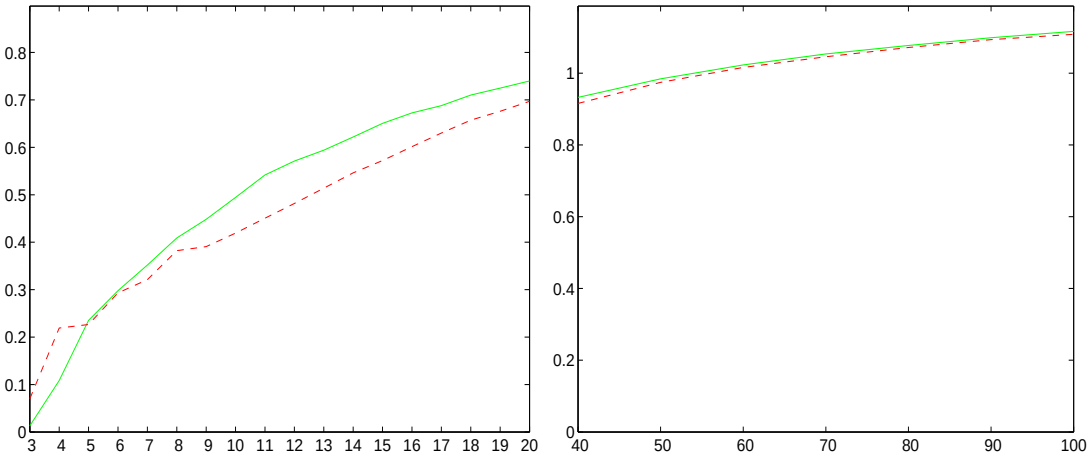


Figure 4.8: Average distances to the closest point on the surface/curve (or on its reflection through the origin). Red dashed: Wegman's Curves. Green solid: our surfaces.

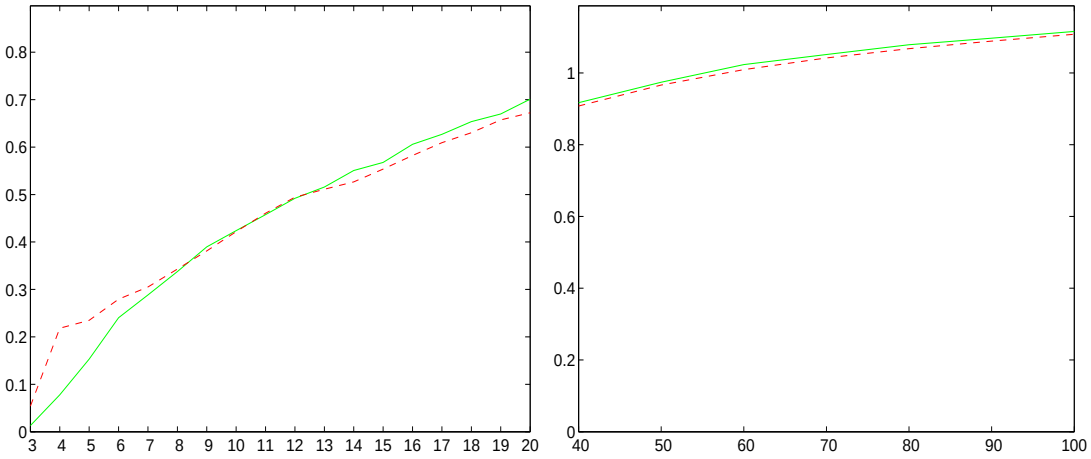


Figure 4.9: Average distances to closest point on surface/curve using λ and μ values obtained from the sequence: $(2^{\sqrt{2}}, 3^{\sqrt{2}}, 5^{\sqrt{2}}, 7^{\sqrt{2}}, 11^{\sqrt{2}}, \ldots)$. Red dashed: Wegman's curves. Green solid: our surfaces.

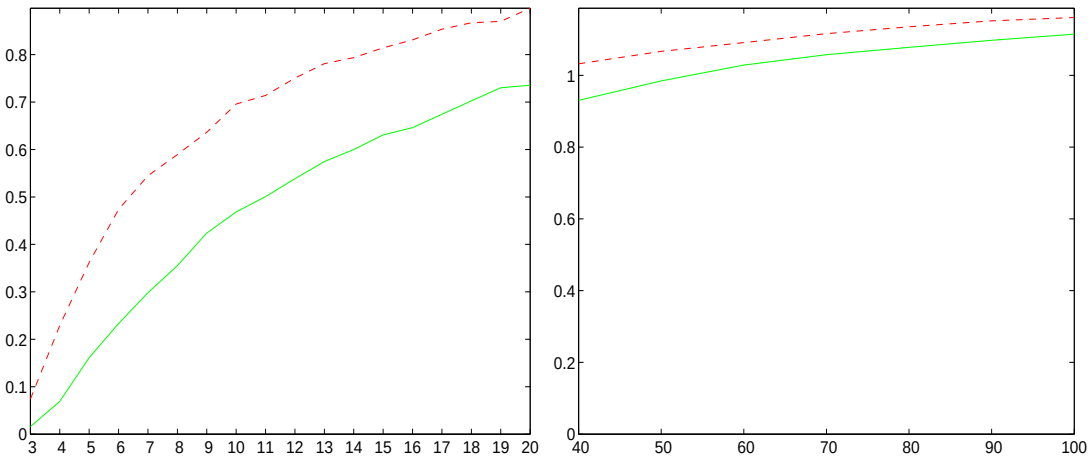


Figure 4.10: Average distances to closest point on surface/curve using λ and μ values obtained from the sequence: $(\pi + 1, \pi + 2, \pi + 3, \pi + 4, \ldots)$. Red dashed: Wegman's curves. Green solid: our surfaces.

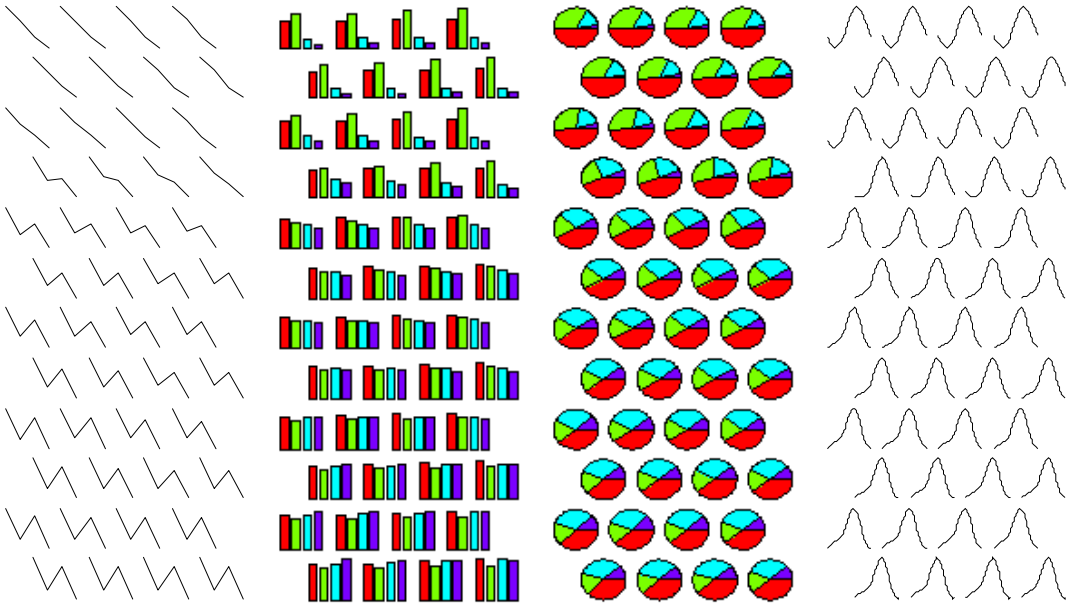


Figure 5.1: Different ways of visualizing the model vectors. The first three are in the existing literature. The fourth uses Andrews' curves. All are representing the model vectors of a SOM trained with the iris data set.

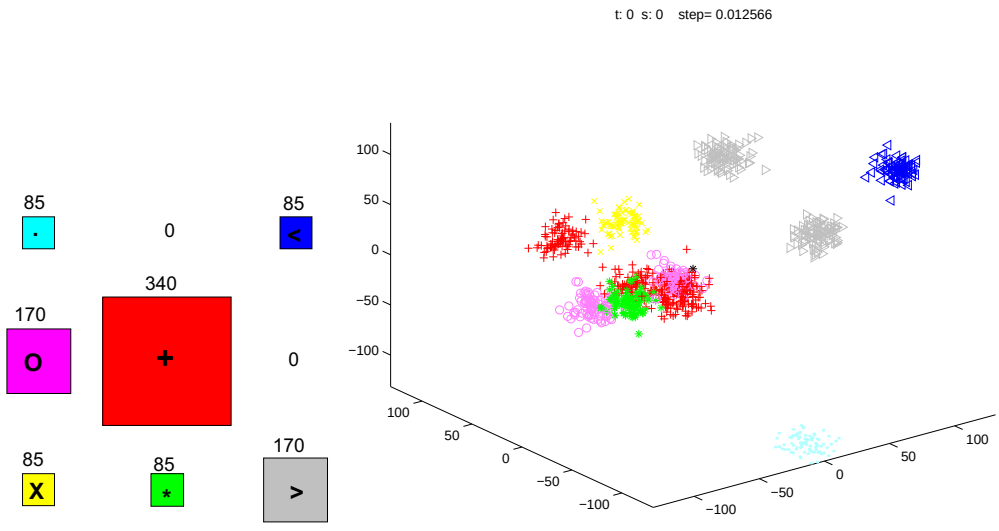


Figure 5.3: Left: the number of times each neuron is the BMU. Right: a projection of the data set using our surfaces.

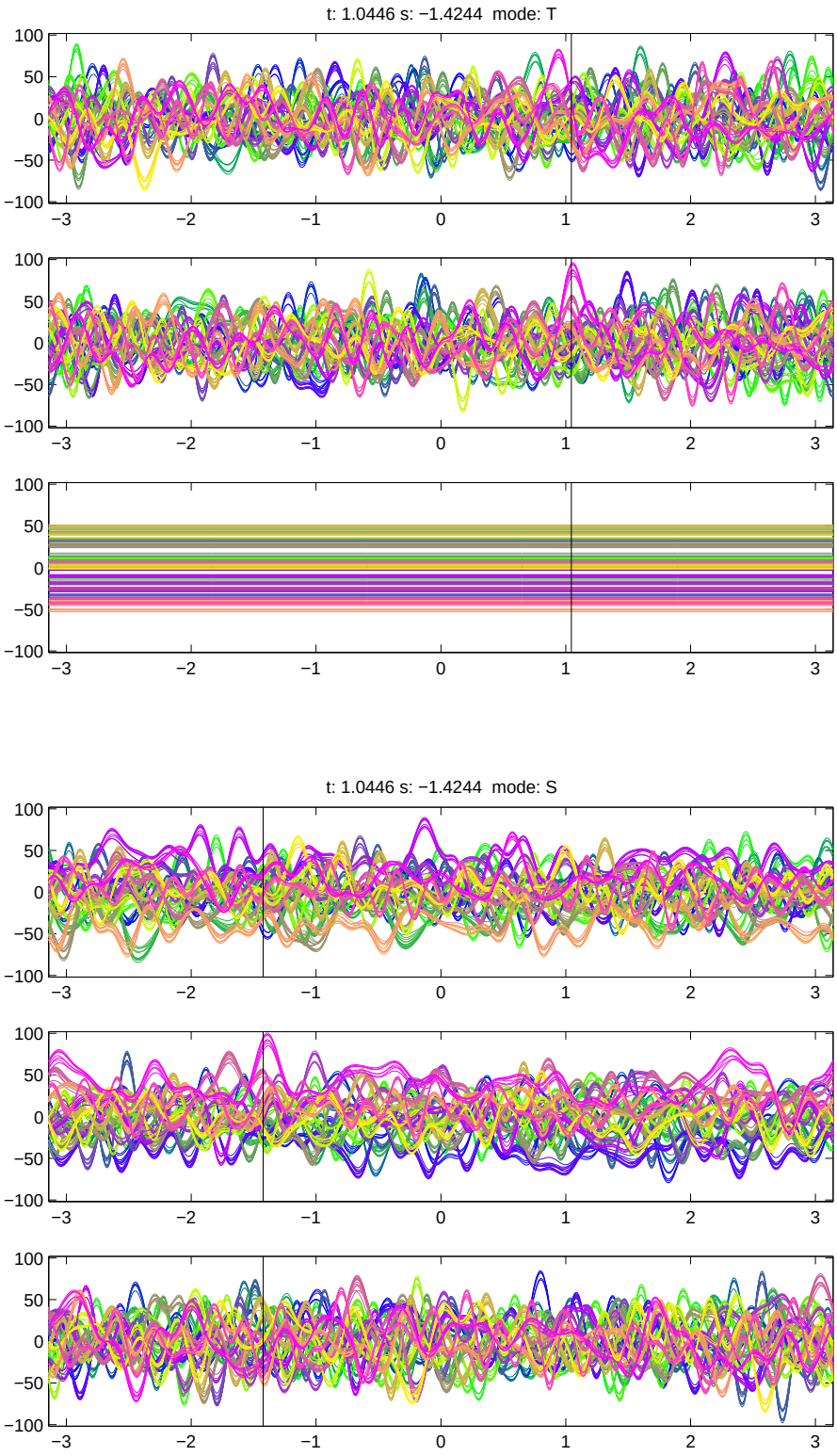


Figure 5.6: Top: T-slice at $s = -1.4244$. Bottom: S-slice at $t = 1.0446$. Structure is becoming visible in terms of clusters of similar data points.

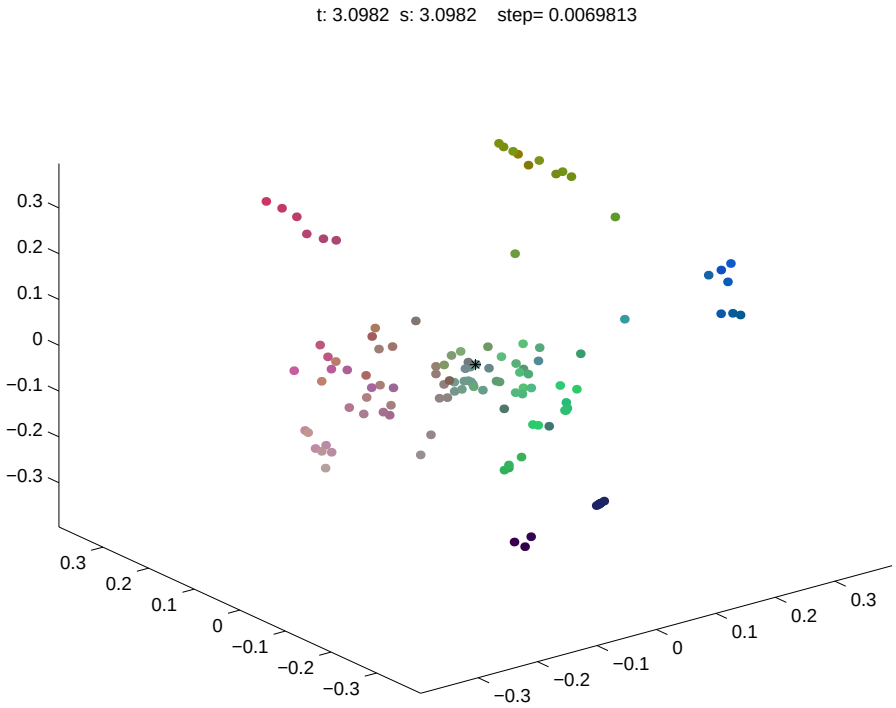


Figure 5.7: A snapshot of a 3D grand tour combined with the auto-colouring of the points (the full animation can be downloaded from `http://cis.paisley.ac.uk/fyfe-ci0/cgo/VIIP2004/GT3D.avi`). The position of the points is obtained from the surfaces constructed using the raw data, the colour comes from the surfaces constructed using the principal component projection of the data.

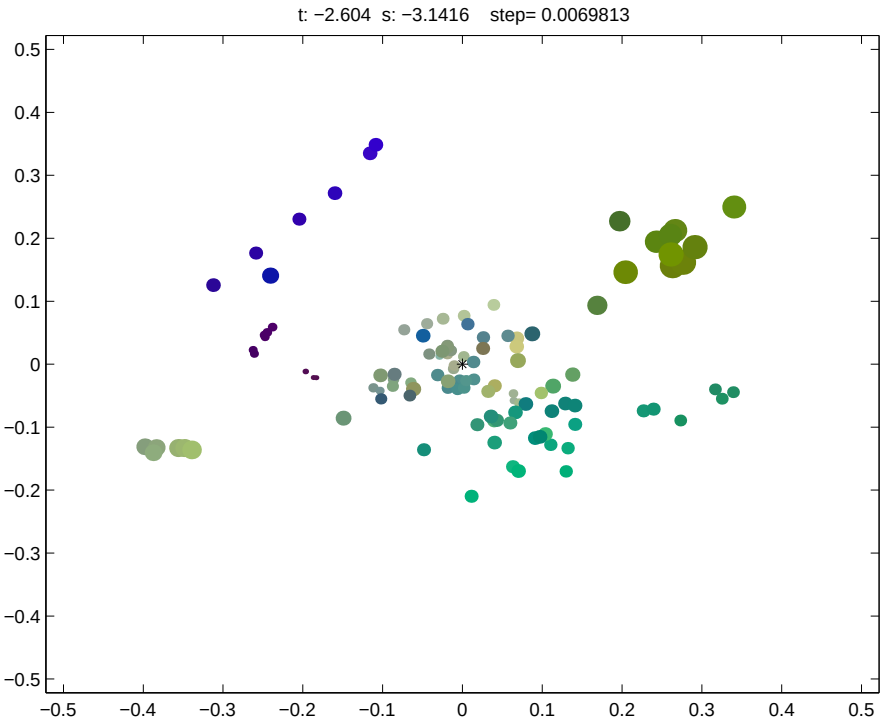


Figure 5.8: A snapshot of a 2D/size grand tour combined with the auto-colouring of the points (the full animation can be downloaded from `http://cis.paisley.ac.uk/fyfe-ci0/cgo/VIIP2004/GT2D.avi`). The position and size of the points come from the raw data, the colour from the principal component projection of the data.

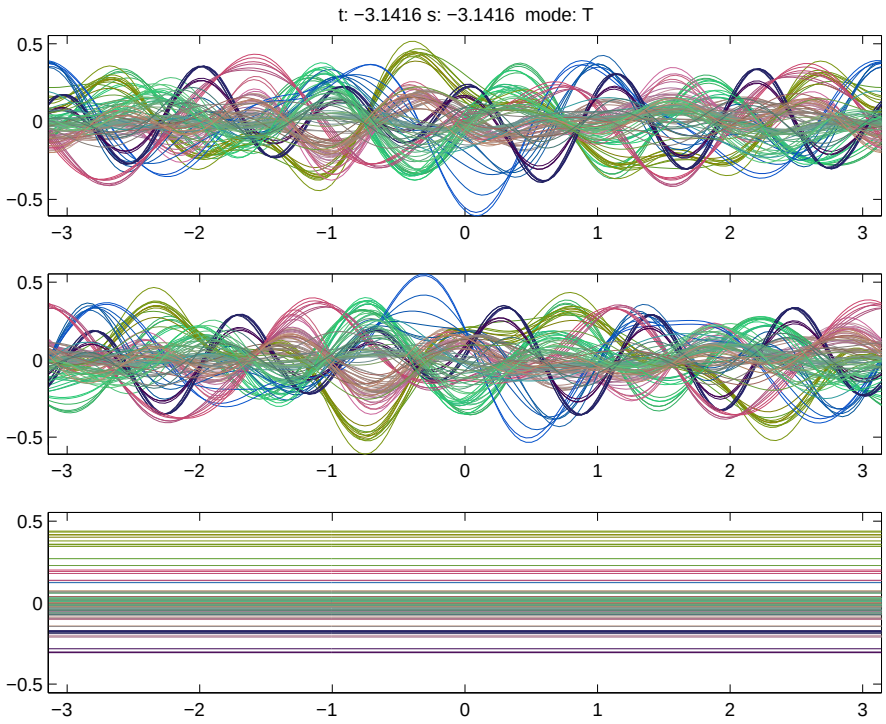


Figure 5.9: A snapshot of a n-D grand tour combined with the *auto-colouring* of the curves obtained from the principal components (the full animation can be downloaded from `http://cis.paisley.ac.uk/fyfe-ci0/cgo/VIIP2004/GTnD.avi`). This can save us having to brush some of the clusters.

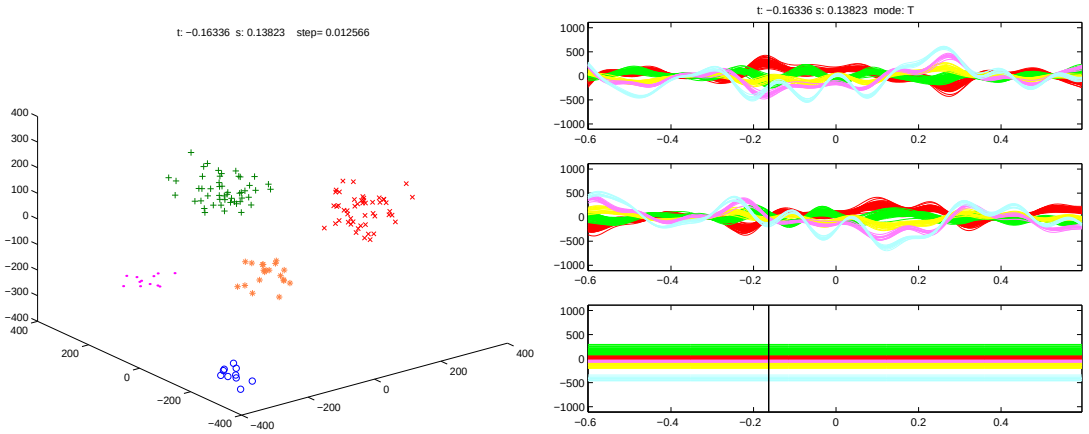


Figure 6.1: Left: a projection of the points of the five clusters using a linear kernel. Right: the T-slice of our surface, at $s = 0.13823$, corresponding to the projection of the figure on the left.

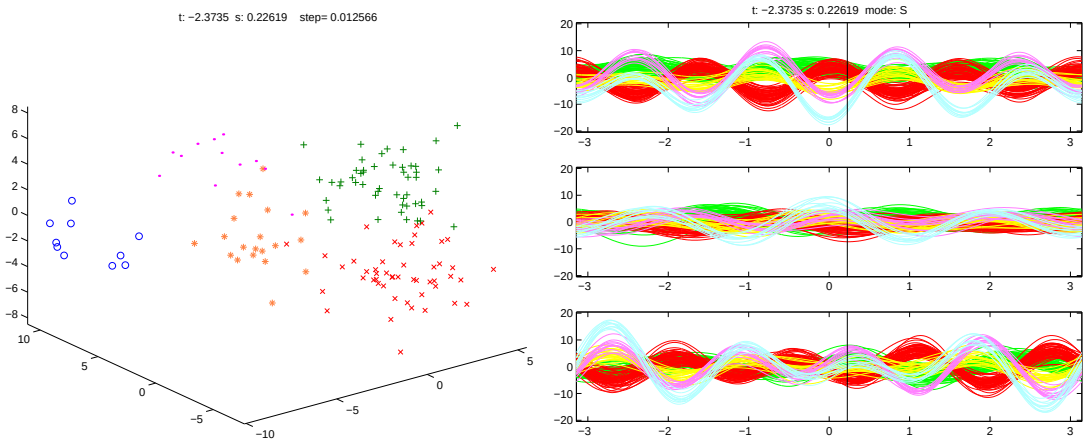


Figure 6.2: Left: a projection of the points of the five clusters. Right: the S-slice of our surface, at $t = -2.3735$, corresponding to the projection of the figure on the left.

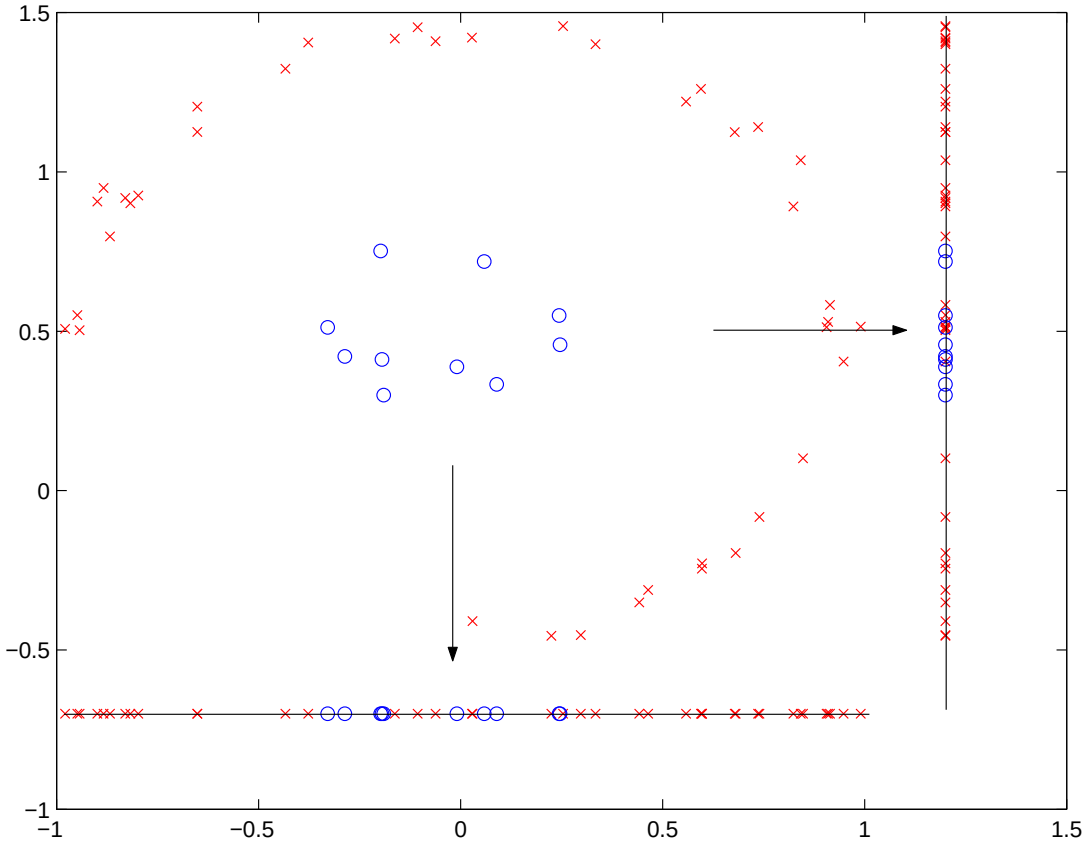


Figure 6.3: Two one-dimensional projections of a two-dimensional data set.

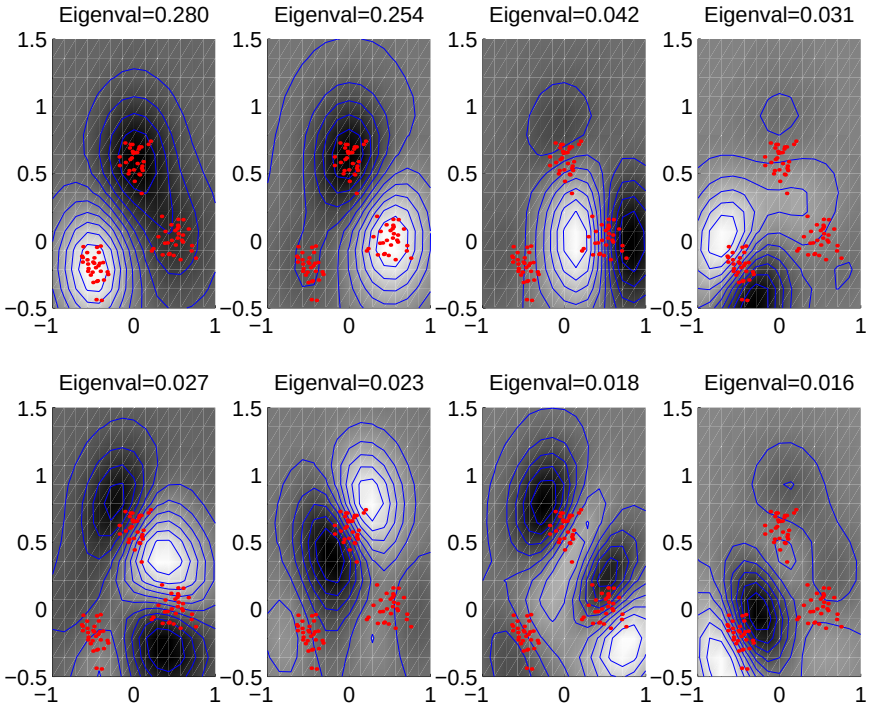


Figure 6.7: The 3 clusters data set is shown as individual points. The contours are contours of equal projection on the respective Principal Components. The first two principal components are sufficient to differentiate between the three clusters; the others slice the clusters internally and have much less variance associated with them.

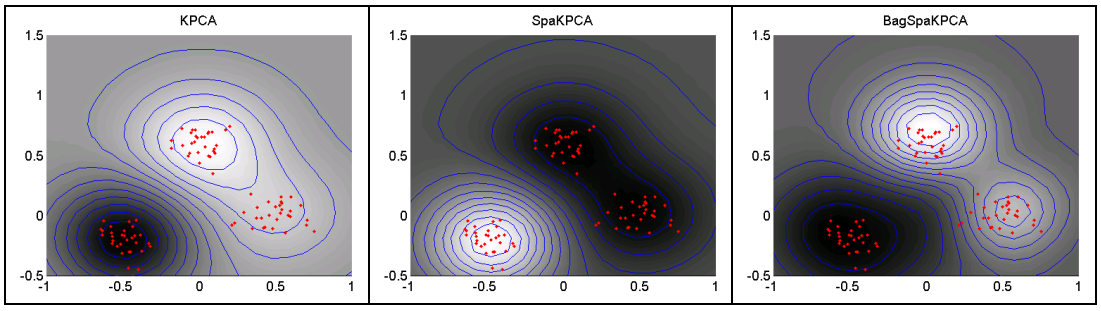


Figure 6.9: First principal component. Left: using the ordinary Kernel PCA. Center: using the Sparse Kernel PCA. Right: using the Bagged Sparse Kernel PCA.

Bibliography

[1] B. Alpern and L. Carter. The Hyperbox. In G. M. Nielson and L. Rosenblum, editors, *Proceedings of IEEE Visualization '91*, pages 133–139, San Diego, California, October 1991.

[2] D. F. Andrews. Plots of High Dimensional Data. *Biometrics*, 28:125–136, 1972.

[3] T. S. Apaiwongse. Facial Display of Environmental Policy Uncertainty. *Journal of Business Psychology*, 10:65–74, 1995.

[4] D. Asimov. The Grand Tour: a Tool for Viewing Multidimensional Data. *SIAM Journal on Scientific and Statistical Computing*, 6(1):128–143, 1985.

[5] D. A. Asimov and A. Buja. Grand Tour via Geodesic Interpolation of 2-frames. In R. J. Moorhead II, D. E. Silver, and S. P. Uselton, editors, *Proceedings of SPIE – Volume 2178 Visual Data Exploration and Analysis*, pages 145–153, 1994.

[6] ATKOSoft S.A. Survey on Visualisation Methods and Software Tools, 1997. `http://europa.eu.int/en/comm/eurostat/research/supcom.96/30/result/a/vi%sualisation_methods.pdf`.

[7] F. Azuaje. A Cluster Validity Framework for Genome Expression Data. *Bioinformatics*, 18:319–320, 2002.

[8] R. A. Becker and W. S. Cleveland. Brushing Scatterplots. *Technometrics*, 29:127–142, 1987.

[9] R. A. Becker, W. S. Cleveland, and A. R. Wilks. Dynamic Graphics for Data Analysis. *Statistical Science*, 2(4):355–383, November 1987.

[10] C. G. Behsers and S. K. Feiner. Visualizing n-Dimensional Virtual Worlds with n-Vision. *Computer Graphics*, 24(2):37–38, 1990.

[11] J. Bertin. *Graphics and Graphic Information Processing.* Walter De Gruyter, Berlin, Germany, 1981.

[12] J. Bertin. *Semiology of Graphics.* University of Wiscosin Press, Madison, WI, 1983.

[13] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford:Clarendon Press, 1995.

[14] N. Bolshakova. Cluster Validity Algorithms. `http://www.cs.tcd.ie/Nadia.Bolshakova/validation_algorithms.html`, 2004. (last visited 31-March-2004).

[15] N. Bolshakova and F. Azuaje. Cluster Validation Techniques for Genome Expression Data. *Signal Processing*, 83(4):825–833, 2003.

[16] L. Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996.

[17] L. Breiman. Using Adaptive Bagging to Debias Regressions. Technical Report 547, Statistics Department, University of California at Berkeley, 1999.

[18] A. Buja and D. Asimov. Grand Tour Methods: An Outline. In D. Allen, editor, *Computer Science and Statistics: Proceedings of the Seventeenth Symposium on the Interface*, pages 63–67, Amsterdam: North Holland, 1986. Elsevier Sicence Publisher B.V.

[19] C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):1–43, 1998.

[20] M. Chalmers. A Linear Iteration Time Layout Algorithm for Visualising High-Dimensional Data. In *Proc. IEEE Visualization 96*, pages 127–132, San Francisco, Oct.-Nov. 1996.

[21] J. M. Chambers, W. S. Cleveland, B. Keliner, and P. A. Tukey. *Graphical Methods for Data Analysis.* Chapman and Hall, New York, 1983.

[22] D. Charles and C. Fyfe. Modelling Multiple Cause Structure using Rectification Constraints. *Network: Computation in Neural Systems*, 9:167–182, May 1998.

[23] H. Chernoff. The Use of Faces to Represent Points in k-dimensional Space Graphically. *Journal of American Statistical Association*, 68:361–368, 1973.

[24] H. Chernoff and M. H. Rizvi. Effect on Classification Error of Random Permutations of Features in Representing Multivariate Data by Faces. *Journal of American Statistical Association*, 70:548–554, 1975.

[25] M. C. Chuah and S. G. Eick. Information Rich Glyphs for Software Management Data. *IEEE Computer Graphics and Applications*, 18(4):24–29, July 1998.

[26] W. S. Cleveland and M. E. McGill, editors. *Dynamic Graphics for Statistics*. Wadsworth & Brooks, Belmont, California, 1988.

[27] E. Corchado and C. Fyfe. Maximum Likelihood Hebbian Learning. In *Tenth European Symposium on Artificial Neural Networks, ESANN2002*, pages 143–148. d-side publications, 2002.

[28] E. Corchado, D. MacDonald, and C. Fyfe. Maximum and Minimum Likelihood Hebbian Learning for Exploratory Projection Pursuit. *Data Mining and Knowledge Discovery*, 8:203–225, 2004.

[29] S. L. Crawford and T. C. Fall. Projection Pursuit Techniques for Visualizing High-dimensional Data Sets. In G. M. Nielson, B. Shrivers, and L. J. Rosenblum, editors, *Visualization in Scientific Computing*, pages 94–108. IEEE Computer Society Press, Los Alamitos, CA, 1990.

[30] D. L. Davies and D. W. Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(4):224–227, 1979.

[31] D. de Ridder and R. P. W. Duin. Sammon's Mapping Using Neural Networks: A Comparison. *Pattern Recognition Letters*, 18(11–13):1307–1316, November 1997.

[32] P. Demartines and J. Hérault. Vector Quantization and Projection Neural Network. In J. Mira, J. Cabestany, and A. Prieto, editors, *International*

Workshop on Artificial Neural Networks, volume 686 of *Lecture Notes in Computer Science*, pages 328–333. Springer Verlag, 1993.

[33] P. Demartines and J. Hérault. Curvilinear Component Analysis: A Self-Organizing Neural Network for Nonlinear Mapping of Data Sets. *IEEE Transactions on Neural Networks*, 8(1):148–154, January 1997.

[34] P. Demartines and J. Hérault. Curvilinear Component Analysis: A Self-Organizing Neural Network for Nonlinear Mapping of Data Sets. *IEEE Transaction on Neural Networks*, 8(1):148–154, January 1997.

[35] G. Deoeck and T. Kohonen. *Visual exploration in Finance using Self-Organizing Maps*. Springer-Verlag, 1998.

[36] P. Diaconis and D. Freedman. Asymptotics of Graphical Projections. *The Annals of Statistics*, 12(3):793–815, 1984.

[37] K. I. Diamantaras and S. Y. Kung. *Principal Component Neural Networks. Theory and Applications*. John Wiley & Sons, 1996.

[38] J. C. Dunn. Well Separated Clusters and Optimal Fuzzy Partitions. *Journal of Cybernetics*, 4:95–104, 1974.

[39] R. Durbin and D. Willshaw. An Analogue Approach to the Traveling Salesman Problem Using an Elastic Net Method. *Nature*, 326(6114):689–691, Apr. 16 1987.

[40] R. Dybowski and S. Roberts. Confidence Intervals and Prediction Intervals for Feed-Forward Neural Networks. In R. Dybowski and V. Gant, editors, *Clinical Applications of Artificial Neural Networks*, pages 298–326. Cambridge University Press, Cambridge, 2001.

[41] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, London, 1993.

[42] P. Embrechts and A. M. Herzberg. Variations of Andrews' Plots. *International Statistical Review*, 59(2):175–194, 1991.

[43] P. Embrechts, A. M. Herzberg, H. K. Kalbfleisch, W. N. Traves, and J. R. Whitla. An Introduction to Wavelets with Applications to Andrews' Plots. *Journal of Computational and Applied Mathematics*, 64:41–56, 1995.

[44] P. Embrechts, A. M. Herzberg, and A. C. Ng. An Investigation of Andrews' Plots to Detect Period and Outliers in Time Series Data. *Communications in Statistics – Simulation and Computation*, 15(4):1027–1051, 1986.

[45] M. C. Ferreira de Oliveira and H. Levkowitz. From Visual Data Exploration to Visual Data Mining: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):378–394, July–September 2003.

[46] B. Flury and H. Riedwyl. Graphical Representation of Multivariate Data by Means of Asymmetrical Faces. *Journal of American Statistical Association*, 76:757–765, 1981.

[47] J. H. Friedman. Exploratory Projection Pursuit. *Journal of the American Statistical Association*, 82(397):249–266, March 1987.

[48] J. H. Friedman and J. W. Tukey. A Projection Pursuit Algorithm for Exploratory Data Analysis. *IEEE Transactions on Computers*, c-23(9):881–889, Sept 1974.

[49] T. Fruchterman and E. Reingold. Graph Drawing by Force-Directed Placement. *Software—Practice and Experience*, 21(11):1129–1164, 1991.

[50] Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner. Hierarchical Parallel Coordinates for Exploration of Large Datasets. In *Proceedings of the 10th IEEE Visualization 1999 Conference (VIS '99): celebrating ten years*, pages 43–50, San Francisco, California, United States, 1999. SIGGRAPH: ACM Special Interest Group on Computer Graphics and Interactive Techniques, IEEE Computer Society Press.

[51] C. Fyfe. PCA Properties of Interneurons. In *From Neurobiology to Real World Computing, ICANN 93*, pages 183–188, 1993.

[52] C. Fyfe. Introducing Asymmetry into Interneuron Learning. *Neural Computation*, 7(6):1167–1181, 1995.

[53] C. Fyfe. Radial Feature Mapping. In *International Conference on Artificial Neural Networks, ICANN95*, volume 2 of *Neuronimes '95 Scientific Conference*, pages 27–32, Paris, France, 1995.

[54] C. Fyfe. A Comparative Study of Two Neural Methods of Exploratory Projection Pursuit. *Neural Networks*, 10(2):257–262, 1997.

[55] C. Fyfe and R. Baddeley. Non-Linear Data Structure Extraction using Simple Hebbian Networks. *Biological Cybernetics*, 72(6):533–541, 1995.

[56] C. Fyfe, R. Baddeley, and D. McGregor. Exploratory projection pursuit: An artificial neural network approach. Technical Report /94/160, The University of Strathclyde, October 1994.

[57] C. Fyfe, R. Baddeley, and D. McGregor. Exploratory Projection Pursuit: An Artificial Neural Network Approach. Technical Report 94/160, University of Strathclyde, 1994.

[58] C. Fyfe and D. Charles. Using Noise to Form a Minimal Overcomplete Basis. In *Seventh International Conference on Artificial Neural Networks, ICANN99*, pages 708–713, 1999.

[59] C. Fyfe and D. MacDonald. Epsilon-insensitive Hebbian Learning. *Neurocomputing*, 47:35–57, 2002.

[60] M. Gallagher. *Multi-layer Perceptron Error Surfaces: Visualization, Structure and Modelling*. PhD thesis, Dept. Computer Science and Electrical Engineering, University of Queensland, 2000.

[61] C. García-Osorio, E. Corchado, and C. Fyfe. Three Neural Exploratory Projection Pursuit Algorithms as Agent Information Providers. In *The International Workshop on the Practical Applications of Agents and Multiagent Systems*, pages 217–226. Universidad de Salamanca, 2002.

[62] C. García-Osorio and C. Fyfe. An Extension of Andrews Curves for Data Analysis. In *Emergent Solutions for the Information and Knowledge Economy (X SIGEF Congress)*, volume 1, pages 123–134, 2003.

[63] C. García-Osorio and C. Fyfe. Initialising Exploratory Projection Pursuit Networks. In M. H. Hamza, editor, *3rd Annual IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP 2003)*, pages 124–128. The International Association of Science and Technology for Development, 2003.

[64] C. García-Osorio and C. Fyfe. Three Neural Exploratory Projection Pursuit Algorithms. In *European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE 2003)*, pages 409–420, 2003.

[65] C. García-Osorio and C. Fyfe. Visualisation in High Dimensional Feature Spaces. In *International Workshop on Practical Applications of Agents and Multiagents Systems (IWPAAMS 2003)*, pages 158–167, Valladolid (Spain), 2003.

[66] C. García-Osorio and C. Fyfe. An Extension of Grand Tour Methods Based on Andrews' Curves. In J. J. Villanueva, editor, *4^{th} Annual IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP 2004)*, pages 109–114, Marbella, Spain, September 6–8 2004.

[67] C. García-Osorio and C. Fyfe. Comparing Exploratory Projection Pursuit Artificial Neural Networks. In V. Botty and E. Corchado, editors, *3^{rd} International Workshop on Practical Applications of Agents and Multiagent Systems, IWPAAMS 2004*, pages 201–210. Universidad de Burgos, 2004.

[68] C. García-Osorio and C. Fyfe. Making Agent's Decisions Transparent. In V. Botty and E. Corchado, editors, *3^{rd} International Workshop on Practical Applications of Agents and Multiagent Systems, IWPAAMS 2004*, pages 339–342. Universidad de Burgos, 2004.

[69] C. García-Osorio, J. Maudes, and C. Fyfe. Using Andrews' Curves for Clustering and Sub-Clustering Self-Organizing Maps. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN 2004)*, pages 477–482, Bruges, Belgium, April 2004. d-side publications.

[70] R. Gnanadesikan. *Methods for Statistical Data Analysis of Multivariate Observations*. John Wiley & son, New York, 1977.

[71] N. A. Goodchild and K. Vijayan. Significance Tests in Plots of Multidimensional Data in two Dimensions. *Biometrics*, 30:209–210, 1974.

[72] L. Goodman and W. Kruskal. Measures of Associations for Cross-Validations. *Journal of American Statistics Association*, 49:732–764, 1954.

[73] A. N. Gorban and A. Y. Zinovyev. Visualization of Data by Method of Elastic Maps and its Applications in Genomics, Economics and Sociology. Technical Report IHES M/01/36, Institut des Hautes Etudes Scientifiques, 2001.

[74] M. Graham and J. Kennedy. Using Curves to Enhance Parallel Coordinate Visualisations. In *Proceedings of the Seventh International Conference on Information Visualization (IV'2003)*, pages 10–16, London, UK, 2003. IEEE Computer Society.

[75] G. Grinstein, R. Pickett, and M. G. Williams. EXVIS: An Exploratory Visualization Environment. In *Proceedings of Graphics Interface '89*, pages 254–259, 1989.

[76] C. G. Hamner, D. W. Turner, and D. M. Young. Comparison of Several Graphical Methods for Representing Multivariate Data. *Computer and Mathematics with Applications*, 13:647–655, 1987.

[77] T. J. Hastie and W. Stuetzle. Principal Curves. *Journal of the American Statistical Association*, 84(406):502–516, June 1989.

[78] H. Hauser, F. Ledermann, and H. Doleisch. Angular Brushing of Extended Parallel Coordinates. In *IEEE Symposium on Information Visualization 2002 (InfoVis 2002)*, pages 127–130, Boston, MA, October 2002. IEEE Computer Society Press.

[79] J. Himberg. Enhancing the SOM-based Data Visualization by Linking Different Data Projections. In *Proceedings of 1st International Symposium IDEAL'98, Intelligent Data Engineering and Learning—Perspectives on Financial Engineering and Data Mining*, pages 427–434. Springer, Hong Kong, 1998.

[80] H. Hotelling. Analysis of a Complex of Statistical Variables into Principal Components. *Journal of Educational Psychology*, 24:417–441, 498–520, 1933.

[81] P. J. Huber. Projection Pursuit. *Annals of Statistics*, 13:435–475, 1985.

[82] L. J. Hubert and J. R. Levin. A General Statistical Framework for Assessing Categorical Clustering in Free Recall. *Psychological Bulleting*, 83:1072–1080, 1976.

[83] M. Y. Huh and K. Kim. Visualization of Multidimensional Data Using Modifications of the Grand Tour. *Journal of Applied Statistics*, 29(5):721–728, July 2002.

[84] A. Hyvarinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley, 2001.

[85] A. Inselberg. Parallel coordinates — how it happened. `http://www.math.tau.ac.il/~aiisreal/` (last visited 5/August/2004).

[86] A. Inselberg. The Plane with Parallel Coordinates. *The Visual Computer*, 1:69–91, 1985.

[87] A. Inselberg. Visualization and Data Mining of High-dimensional Data. *Chemometrics and Intelligent Laboratory Systems*, 60(1-2):147–159, January 2002.

[88] A. Inselberg, Y. Chen, M. Shieh, and H. Lee. Planar Conflict Resolution Algorithms for Air Traffic Control. In *2nd Canadian Conference on Computational Geometry*, pages 160–164, 1990.

[89] A. Inselberg and B. Dimsdale. Parallel Coordinates: A Tool for Visualizing Multidimensional Geometry. In *Proceedings of Visualization '90*, pages 361–378, Los Alamitos, CA, USA, 1990. SIGGRAPH: ACM Special Interest Group on Computer Graphics and Interactive Techniques, Publisher IEEE Computer Society Press.

[90] I. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[91] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[92] M. C. Jones and R. Sibson. What is Projection Pursuit. *Journal of The Royal Statistical Society*, pages 1–37, 1987.

[93] J. Karhunen and J. Joutsensalo. Representation and separation of signals using nonlinear PCA type learning. *Neural Networks*, 7(1):113–127, 1994.

[94] S. Kaski. *Data Exploration Using Self-Organizing Maps*. PhD thesis, Helsinki University of Technology, 1997.

[95] S. Kaski, J. Venna, and T. Kohonen. Coloring that Reveals Cluster Structures in Multivariate Data. *Australian Journal of Intelligent Information Processing Systems*, 6:82–88, 2000.

[96] D. A. Keim. Visual Techniques for Exploring Databases, Invited Tutorial. In *Int. Conference on Knowledge Discovery in Databases (KDD'97)*, Newport Beach, CA, 1997. `http://www.dbs.informatik.uni-muenchen.de/~daniel/KDD97.pdf` (last visited 28/June/2004).

[97] D. A. Keim. Designing Pixel-Oriented Visualization Techniques: Theory and Applications. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):59–72, January-March 2000.

[98] D. A. Keim. Information Visualization and Visual Data Mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, January-March 2002.

[99] D. A. Keim and H.-P. Kriegel. VisDB: Dabatase Exploration Using Multidimensional Visualization. *IEEE Computer Graphics and Applications*, 14(5):44–49, September 1994.

[100] D. A. Keim and H.-P. Kriegel. Visualization Techniques for Mining Large Databases: A Comparison. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):923–938, December 1996.

[101] R. Khattree and D. N. Naik. Andrews Plots for Multivariate Data: some new Suggestions and Applications. *Journal of Statistical Planning and Inference*, 100:411–425, 2002.

[102] B. Kleiner and J. Hartigan. Representing Points in many Dimension by Trees and Castles. *Journal of American Statistical Association*, 76:260–269, 1981.

[103] T. Kohonen. Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics*, 43:59–69, 1992.

[104] T. Kohonen. *Self-Organizing Maps*. Springer, 3rd edition, 2001.

[105] J. A. Kokiol and W. Hacke. A Bivariate Version of Andrews Plots. *IEEE Transactions on Biomedical Engineering*, 38(12):1271–1274, 1991.

[106] J. B. Kruskal and M. Wish. *Multidimensional Scaling*. Sage Publications, Beverly Hills, California, 1978.

[107] S. R. Kulkarmi and S. R. Paranjape. Use of Andrews' Function Plot Technique to Construct Control Curves for Multivariate Process. *Communications in Statistics – Theory Methods*, 13(20):2511–2533, 1984.

[108] J. LeBlanc, M. O. Ward, and N. Wittels. Exploring N-Dimensional Databases. In A. Kaufman, editor, *Proceedings of the 1st conference on Visualization '90*, pages 230–237, San Francisco, California, October 1990. IEEE Computer Society Technical Committee on Computer Graphics, IEEE Computer Society Press.

[109] J. A. Lee, C. Archambeau, and M. Verleysen. Locally Linear Embedding versus Isotop. In *ESANN'2003 proceedings - European Symposium on Artificial Neural Networks*, pages 527–534. d-side publications, 2003.

[110] J. A. Lee, A. Lendasse, N. Donckers, and M. Verleysen. A Robust Nonlinear Projection Method. In M. Verleysen, editor, *ESANN'2000 proceedings - European Symposium on Artificial Neural Networks*, pages 13–20. D-Facto publications, 2000.

[111] J. A. Lee, A. Lendasse, and M. Verleysen. Curvilinear Distance Analysis versus Isomap. In M. Verleysen, editor, *ESANN'2002 proceedings - European Symposium on Artificial Neural Networks*, pages 185–192. d-side publications, 2002.

[112] J. A. Lee and M. Verleysen. Nonlinear Projection with the Isotop Method. In J. R. Dorronsoro, editor, *ICANN 2002 proceedings - International Conference on Artificial Networks, Madrid (Spain)*, volume 2415 of *Lecture Notes in Computer Science*, pages 933–938. Springer, 2002.

[113] M. D. Lee, R. E. Reilly, and M. E. Butavicius. An Empirical Evaluation of Chernoff Faces, Star Glyphs, and Spatial Visualizations for Binary Data. In *Proceedings of the Australian symposium on Information visualisation*, pages 1–10. Australian Computer Society, Inc., 2003.

[114] Y. K. Leung and M. D. Apperley. A Review and Taxonomy of Distortion-Oriented Presentation Techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, 1994.

[115] H. Levkowitz. Color Icons: Merging Color and Texture Perception for Integrated Visualization of Multiple Parameters. In G. M. Nielson and L. Rosen-

blum, editors, *Proceedings of IEEE Visualization '91*, pages 164–170, San Diego, California, October 1991. SIGGRAPH: ACM Special Interest Group on Computer Graphics and Interactive Techniques, IEEE Computer Society Press.

[116] M. S. Lewis-Beck, editor. *Factor Analysis and Related Techniques*. Sage Publications, London, 1994.

[117] A. Loizides and M. Slater. The Empathic Visualisation Algorithm (EVA) – An Automatic Mapping from Abstract Data to Naturalistic Visual Structure. In *Sixth International Conference on Information Visualization (IV'02)*, pages 705–712, 2002.

[118] J. A. Lott and T. C. Durbridge. Use of Chernoff Faces to Follow Trends in Laboratory Data. *Journal of Clinical Laboratory Analysis*, 4(1):59–63, 1990.

[119] D. MacDonald and C. Fyfe. Data Mining using Unsupervised Neural Networks. In *The Third International Conference on Soft Computing, SOCO99*, pages 173–178, 1999.

[120] J. Mao and J. A. K. Artificial Neural Networks for Feature Extraction and Multivariate Data Projection. *IEEE Transaction on Neural Networks*, 6(2):296–317, March 1995.

[121] S. Mika, B. Scholkopf, A. Smola, K.-R. Muller, M. Scholz, and G. Ratsch. Kernel PCA and de-Noising in Feature Spaces. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Processing Systems, 11*, pages 536–542. MIT Press, 1999.

[122] S. Moriarity. Communicating Financial Information through Multidimensional Graphics. *Journal of Accounting Research*, 17(1):205–224, 1979.

[123] A. Morrison and M. Chalmers. Improving Hybrid MDS with Pivot-Based Searching. In *Proc. IEEE Information Visualization 2003*, pages 85–90, Seattle, 2003.

[124] A. Morrison, G. Ross, and M. Chalmers. A Hybrid Layout Algorithm for Sub-Quadratic Multidimensional Scaling. In *Proc. IEEE Information Visualisation 2002*, pages 152–160, Boston, October 2002.

[125] A. Morrison, G. Ross, and M. Chalmers. Fast Multidimensional Scaling through Sampling, Springs and Interpolation. *Information Visualization*, 2(1):68–77, March 2003.

[126] W. Müller and M. Alexa. Using Morphing for Information Visualization. In *Proceedings of the 1998 workshop on New paradigms in information visualization and manipulation*, pages 49–52. ACM Press, 1998.

[127] J. F. Murphy. *Methods for Collection and Processing of Gene Expression Data*. PhD thesis, California Institute of Technology, Pasadena, California 91125, 2003.

[128] D. Nel, L. Pitt, and T. Webb. Using Chernoff Faces to Portray Service Quality Data. *Journal of Marketing Management*, 10:247–255, 1994.

[129] E. Oja. A Simplified Neuron Model as a Principal Component Analyser. *Journal of Mathematical Biology*, 16:267–273, 1982.

[130] E. Oja. Neural Networks, Principal Components and Subspaces. *International Journal of Neural Systems*, 1:61–68, 1989.

[131] E. Oja, H. Ogawa, and J. Wangviwattana. Principal Component Analysis by Homogeneous Neural Networks, Part 1: The Weighted Subspace Criterion. *IEICE Trans. Inf. & Syst.*, E75-D:366–375, May 1992.

[132] E. Oja, H. Ogawa, and J. Wangviwattana. Principal Component Analysis by Homogeneous Neural Networks, Part 2: Analysis and Extensions of the Learning Algorithms. *IEICE Trans. Inf. & Syst.*, E75-D(3):375–381, May 1992.

[133] E. A. Rietman and N. Layadi. A Study on $\mathbb{R}^m \rightarrow \mathbb{R}^1$ Maps: Application to a 0.16-μm Via Etch Process Endpoint. *IEEE Transactions on Semiconductor Manufacturing*, 13(4):457–468, 2000.

[134] E. A. Rietman, J. T. C. Lee, and N. Layadi. Dynamic Images of Plasma Processes: Use of Fourier Blobs for Endpoint Detection during Plasma Etching of Patterned Wafers. *Journal of Vacuum Science and Technology*, 16(3):1449–1453, 1998.

[135] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press., 1996.

[136] S. Romdhani, S. Gong, and A. Psarrou. A Multi-View Nonlinear Active Shape Model using Kernel PCA. In T. Pridmore and D. Elliman, editors, *10th British Machine Vision Conference*, volume 2, pages 483–492, Nottingham, UK, 1999. BMVA Press.

[137] P. J. Rousseeuw. Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, 20(1):53–65, 1987.

[138] S. T. Roweis and L. K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, Dec. 22 2000.

[139] H. Sagan. *Space-Filling Curves*. Springer-Verlag, New York, 1994.

[140] J. Sammon. A Nonlinear Mapping for Data Structure Analysis. *IEEE Transactions on Computation*, C-18(5):401–409, 1969.

[141] N. Sawasdichai and S. Poggenpohl. User Purposes and Information-seeking Behaviors in Web-based Media: a User-centered Approach to Information Design on Websites. In *Proceedings of the conference on Designing interactive systems*, pages 201–212. ACM Press, 2002.

[142] P. C. Saxena and K. Navaneetham. The Validation of Chernoff Faces as Clustering Algorithm. In *Proceedings of the VIII Annual Conference of the Indian Society for Probability and Statistics*, pages 179–193, Kolhapur, Shivaji University, 1986.

[143] P. C. Saxena and K. Navaneetham. The Effect of Cluster Size, Dimensionality, and Number of Clusters on Recovery of True Cluster Structure through Chernoff-type Faces. *The Statistician*, 40:415–425, 1991.

[144] B. Scholkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Muller, G. Ratsch, and A. J. Smola. Input Space vs Feature Space in Kernel-based Methods. *IEEE Transactions on Neural Networks*, 10:1000–1017, 1999.

[145] B. Scholkopf, S. Mika, A. Smola, G. Ratsch, and K.-R. Muller. Kernel PCA Pattern Reconstruction via Approximate pre-Images. In R. Z. L. Niklasson, M. Boden, editor, *Proceedings of 8th International Conference on Artificial Neural Networks*, pages 147–152. Springer Verlag, 1998.

[146] B. Scholkopf, A. Smola, and K.-R. Muller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Technical Report 44, Max Planck Institut fur biologische Kybernetik, Dec 1996.

[147] B. Scholkopf, A. Smola, and K.-R. Muller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10:1299–1319, 1998.

[148] B. Scholkopf, A. Smola, and K.-R. Muller. Kernel Principal Component Analysis. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 327–370. MIT Press, 1999.

[149] R. N. Shepard, A. K. Romney, and S. B. Nerlove, editors. *Multidimensional Scaling: Theory and Applications in the Behavioral Sciences*, volume 1. Seminar Press, Inc., 1972.

[150] B. Shneiderman. Tree Visualization with Treemaps: A 2D Space-Filling Approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.

[151] B. Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualization. In *Proceedings IEEE Workshop Visual Languages '96*, pages 336–343. IEEE Computer Society Press, 1996.

[152] H. Siirtola. Direct Manipulation of Parallel Coordinates. In J. Roberts, editor, *Proceedings of the International Conference on Information Visualization (IV'2000)*, pages 373–378. IEEE Computer Society, July 2000.

[153] H. Siirtola. Combining Parallel Coordinates with the Reorderable Matrix. In J. Roberts, editor, *Proceedings of the International Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV 2003)*, pages 63–74, London, UK, July 2003. IEEE Computer Society.

[154] M. Smith and R. Taffler. Improving the Communication of Accounting Information through Cartoon Graphics. *Journal of Accounting, Auditing & Accountability*, 9(2):68–85, 1996.

[155] A. J. Smola, O. L. Mangasarian, and B. Scholkopf. Sparse Kernel Feature Analysis. Technical Report 99-04, University of Wiscosin Madison, 1999.

[156] A. J. Smola, S. Mika, B. Scholkopf, and R. C. Williamson. Regularized Principal Maniforlds. *Machine Learning*, pages 1–28, 2000.

[157] N. H. Spencer. Investigating Data with Andrews Plots. *Social Science Computer Review*, 21(2):244–249, 2003.

[158] J. Symanzik, E. J. Wegman, A. J. Braverman, and Q. Luo. New Applications of the Image Grand Tour. *Computing Science and Statistics*, 34:500–512, 2002.

[159] J. B. Tenenbaum. Mapping a Manifold of Perceptual Observations. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10, pages 682–688, Cambridge, MA, 1998. MIT Press.

[160] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, Dec. 22 2000.

[161] V. Tereshko and N. M. Allinson. Common Framework for "Topographic" and "Elastic" Computations. In D. S. Broomhead, E. A. Luchinskaya, P. V. E. McClintock, and T. Mullin, editors, *Stochaos: Stochastic and Chaotic Dynamics in the Lakes*, volume 502 of *AIP Conference Proceedings*, pages 124–129, 2000.

[162] V. Tereshko and N. M. Allinson. Combining Lateral and Elastic Interactions: Topology-Preserving Elastic Nets. *Neural Processing Letters*, 15:213–223, 2002.

[163] V. Tereshko and N. M. Allinson. Theory of Topology-Preserving Elastic Nets. In W. Klonowski, editor, *Attractors, Signals and Synergetics*, EUROATTRACTOR 2000, pages 215–221. PABS Science Publications, 2002.

[164] F. E. Tidmore and D. W. Turner. On Clustering with Chernoff-type Faces. *Communications in Statistics*, A12(14):381–396, 1983.

[165] W. S. Torgerson. Multidimensional Scaling: I. Theory and Methods. *Psychometrika*, 17(4):401–419, December 1952.

[166] E. R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, Connecticut, 1983.

[167] J. Tukey. *Exploratoy Data Analysis.* Addison-Wesley, Reading, MA, 1977.

[168] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, New York, 1995.

[169] J. Vesanto. Data Mining Techniques Based on the Self-Organizing Map. Master's thesis, Helsinki University of Technology, May 1997.

[170] J. Vesanto. SOM-based Data Visualization Methods. *Intelligent-Data-Analysis*, 3:111–26, 1999.

[171] J. Vesanto and E. Alhoniemi. Clustering of the Self-Organizing Map. *IEEE Transaction on Neural Networks*, 11(3):586–600, May 2000.

[172] M. O. Ward. XmdvTool: Integrating Multiple Methods for Visualizing Multivariate Data. In G. M. Nielson and L. Rosenblum, editors, *Proceedings of the conference on Visualization '94*, pages 326–333, Washinton, D.C., 1994. SESSION: Visualization systems table of contents.

[173] C. Ware and J. C. Beatty. Using Color Dimensions to Display Data Dimensions. *Hum. Factors*, 30(2):127–142, 1988.

[174] E. J. Wegman. Hyperdimensional Data Analysis Using Parallel Coordinates. *Journal of the American Statistical Association*, 411(85):664–675, July 1990.

[175] E. J. Wegman. The Grand Tour in k-dimensions. In C. Page and R. LePage, editors, *Computing Science and Statistics: Proceedings of the 22nd Symposium on the Interface*, pages 127–136. Springer-Verlag, 1991.

[176] E. J. Wegman and Q. Luo. Construction of Line Densities for Parallel Coordinate Plots. In A. Buja and P. Tukey, editors, *Computing and Graphics in Statistics*, pages 107–124. Springer-Verlag, New York, NY, 1991.

[177] E. J. Wegman and Q. Luo. High Dimensional Clustering Using Parallel Coordinates and the Grand Tour. *Computing Science and Statistics*, 28:352–360, 1997.

[178] E. J. Wegman, W. L. Poston, and J. L. Solka. Image grand tour. Technical Report TR 150, The Center for Computational Statistics, `ftp://www.galaxy.gmu.edu/pub/papers/Image_Tour.pdf`, April 1998.

[179] E. J. Wegman and J. Shen. Three-Dimensional Andrews Plots and the Grand Tour. *Computing Science and Statistics*, 25:284–288, 1993.

[180] E. J. Wegman and J. L. Solka. On Some Mathematics for Visualising High Dimensional Data. *Indian Journal of Statistics*, 64(Series A, 2):429–452, 2002.

[181] E. J. Wegman and J. L. Solka. On some mathematics for visualising high dimensional data. *Indian Journal of Statistics*, 64(Series A, 2):429–452, 2002.

[182] J. J. v. Wijk and R. v. Liere. Hyperslice Visualization of Scalar Functions of Many Variables. In G. M. Nielson and R. D. Bergeron, editors, *Proceedings IEEE Visualization '93*, pages 119–125, San Jose, California, October 1993.

[183] P. C. Wong and R. D. Bergeron. 30 Years of Multidimensional Multivariate Visualization. In G. M. Nielson, H. Hagan, and H. Muller, editors, *Scientific Visualization — Overviews, Methodologies and Techniques*, pages 3–33. IEEE Computer Society Press, Los Alamitos, California, 1997.

[184] L. Xu. Least Mean Square Error Reconstruction Principle for Self-Organizing Neural-Nets. *Neural Networks*, 6(5):627 – 648, 1993.