Piecewise-Linear Manifold Learning

Harry Strange

Submitted in total fulfilment of the requirements
of the degree of Doctor of Philosophy

October 2011

Department of Computer Science
Aberystwyth University

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed: ____________________________

Date: ________________

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Where correction services have been used, the extent and nature of the correction is clearly marked in a footnote(s).

Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed: ____________________________

Date: ________________

Statement 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed: ____________________________

Date: ________________

To my parents

Acknowledgements

First and foremost I would like to thank my supervisor Prof. Reyer Zwiggelaar for his help and support over the last three years. Without his wise guidance, assistance, and encouragement, this thesis would not have been possible. My thanks also goes to Dr Mark Neal for his work as my second supervisor. His insightful comments and criticisms were invaluable when finishing this work.

I would also like to extend my gratitude to the Vision, Graphics and Visualisation Group for their feedback and advice on various aspects of my research over the last three years. To the Department of Computer Science for their openness, support, and friendliness, I owe gratitude and thanks.

To my friends and family who have supported me during my PhD I would like to add my thanks. Finally, this thesis is done in the name of the Lord Jesus Christ and for His glory. *Sola scriptura, Sola fide, Sola gratia, Solus Christus, Soli Deo gloria.*

Abstract

The need to reduce the dimensionality of a dataset whilst retaining inherent manifold structure is key in many pattern recognition, machine learning and computer vision tasks. This process is often referred to as manifold learning since the structure is preserved during dimensionality reduction by learning the intrinsic low-dimensional manifold that the data lies on. Since the inception of manifold learning much research has been carried out into the most effective way of tackling this problem. Two main streams emerged to tackle the task: local and global methods. Each aim to preserve either local or global properties of the data. However, in recent years a third stream of research has come forth: global alignment of local models, which aims to preserve local properties over a global scale. We present a framework to tackle this local/global problem that approximates the manifold as a set of piecewise linear models (**Piecewise Linear Manifold Learning**). By merging these linear models in an order defined by their global topology, we can obtain a globally stable, and locally accurate model of the manifold. Examining the local properties of the data also allows us to present a generalisation to one of the main open problems in manifold learning — the out-of-sample extension. This problem is concerned with embedding new samples into a previously learnt low-dimensional embedding. Our solution — **GOoSE** — exploits the local geometry of the manifold to project novel samples into a low-dimensional embedding independent of what manifold learning algorithm was initially used. The results obtained for both Piecewise Linear Manifold Learning and GOoSE are significantly improved over existing state of the art algorithms.

Contents

Schönheit ist ein gutes empfehlungsschreiben
German proverb

1 Introduction[1]

This thesis is concerned with the development of a manifold learning algorithm that is easy to understand, yields good results, scalable and is stable under different data conditions and parameter values. As such, the problem of manifold learning is phrased as a piecewise-linear problem, where the manifold is partitioned into small linear pieces which are reconstructed in a piecewise fashion. Out of this algorithm springs the extension to embed new samples without re-learning the entire data — the out-of-sample extension. From the principles contained in this out-of-sample extension method we are able to extend the algorithm to a generalised form meaning that new samples can be embedded into any mapping produce by any manifold learning algorithm.

In this chapter we lay the foundations for this thesis, namely introducing manifold learning and the manifold assumption. These two key ideas lie not only at the heart of manifold learning theory but also at the heart of this thesis. We end the chapter by outlining the structure of the remainder of the thesis.

[1]Harry Strange and Reyer Zwiggelaar. Classification Performance related to Intrinsic Dimensionality in Mammographic Image Analysis. In Proceedings of the Thirteenth Annual Conference on Medical Image Understanding and Analysis, pp. 219-223. 2009.

1.1 Manifold Learning and the Manifold Assumption

High-dimensional data is often hard to interpret and is found in many machine learning, computer vision and pattern recognition problems. For example, consider the case of a set of images of size 32×32 pixels depicting a 3-dimensional object undergoing rotation. Each frame consists of an image of the object at a given rotation angle and the next image in the sequence will increase this rotation angle until the object has been rotated by $360°$. Each of these images are considered in their rasterised form, that is they are a single vector in 1024-dimensional space, this means that if we take an image every $5°$ then we have a set of 72 vectors in 1024-dimensional space. Given the data in this form, one question that arrises is: *from these samples can we simplify and deduce any structure from the data?* In this example it would be expected that the data would contain circular topology and so lies on a 1-dimensional manifold. However, this 1-dimensional manifold can be embedded into 2-dimensional Euclidean space as a circle. This would be expected because the only change between frames is the angular rotation of the object around one axis and the radius of rotation. This idea that the data lies on or near a 1-dimensional manifold is known as the *manifold assumption.* We assume that even though the input data is high-dimensional (in this case 1024-dimensions), it actually lies on or near a low-dimensional manifold (in this example 1-dimension). Therefore we expect a good manifold learning algorithm to be able to learn this intrinsic low-dimensional manifold and so reduce the dimensionality of the data whilst retaining the important underlying structure. Figure 1.1 pictorially shows this simple example by using a set of images taken from the COIL-20 dataset (Nene et al., 1996) depicting a 3-dimensional cat object rotating $360°$ over 72 images. Using manifold learning we can reduce the dimensionality of the data and so describe each object as the (x, y) position on a circle rather than a point in the

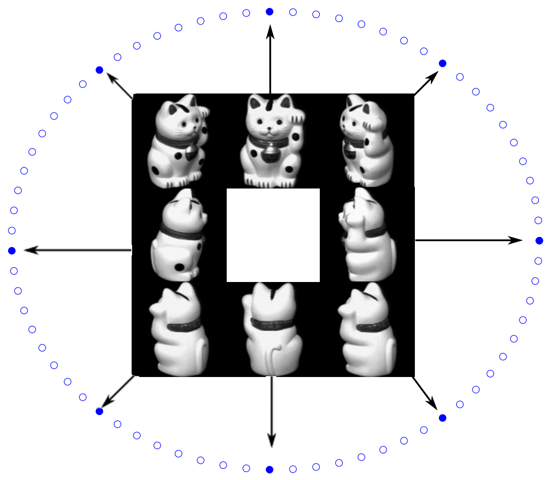


Figure 1.1: An example of a high-dimensional problem that can be expressed in terms of a low-dimensional manifold. The input images are vectors in 1024-dimensional space which can be represented using only 2-dimensions — their position on the circular manifold. Each blue dot represents a single image's position on the learnt low-dimensional manifold.

high-dimensional input space. If we embedded this data set into 1-dimension then there would be a cut in the topology and a discontinuity in the manifold. Thus by reducing the dimensionality we have managed to simplify this rather trivial problem from that of a set of points in 1024-dimensional space to a set of points on a circle in 2-dimensional space. More than that we have also managed to infer some information about the structure of the underlying data. This is best explained by imagining that we had no idea what the input data in this example represented (i.e. we did not know that they were rasterised images of a rotating object).

As mentioned above at the heart of manifold learning is the assumption that the high-dimensional data lies on or near a low-dimensional manifold. In recent years this assumption has been shown to hold both experimentally, through the use of manifold learning on data and visualising the results (e.g. (Tenenbaum et al., 2000; Weinberger and Saul, 2006b; Strange and Zwiggelaar, 2009)); and, in part, theoretically (Seung and Lee, 2000)). Some preliminary work that we performed at the start of this thesis showed, through the use of relating classification performance to intrinsic dimensionality, that mammographic risk assessment can be related to an intrinsic low-dimensional manifold (Strange and Zwiggelaar, 2009). This shows that in classification tasks the manifold structure of the data should not be ignored as it can be used to aid classification. This is not surprising as it has been argued that visual perception is tightly linked to manifold structure (Seung and Lee, 2000). Since mammographic image analysis is a visual problem our preliminary work showed that the link between visual perception and manifold learning exists even in non-trivial problem areas.

The heart of this argument is that manifolds are fundamental to perception, so the brain must have some way of representing them (Seung and Lee, 2000). Because the possible images of an object lie on a manifold, it has been hypothesised that a visual memory is stored as a manifold of stable states or a continuous attractor (Seung and Lee, 2000). Effectively, the fact that manifold learning algorithms show that images can lie on low-dimensional manifolds it can be assumed that the brain must function in a similar way. This leads to an interesting interplay between the theoretical and experimental areas of manifold learning and perception research. The experimental research shows that the images lie on a low-dimensional manifold, and the experiments are based on the claims made by the theoretical research. Similarly the theoretical research claims that the experimental results show that the theoretical foundations are fair assumptions. However the problem is phrased, be it experimental or theoretical, it is still interesting to note the role that manifold learning plays in visual perception and so by extension computer vision, machine learning and pattern recognition.

1.2 Problem Statement

The questions at the heart of this thesis are whether we can develop a manifold learning algorithm that models the data correctly at a local and global scale, and whether the central ideas of this manifold learning algorithm can be extended to handle new samples without the need to re-learn the entire training data. As such the work in this thesis is concerned with two main areas of manifold learning: the development of a manifold learning algorithm, Piecewise Linear Manifold Learning, and the creation of a mapping function that embeds new samples between high-dimensional spaces and previously learnt embeddings, the Generalised Out-of-Sample Extension algorithm. The connection between these two algorithms is described as follows.

Manifold learning seeks to find a low dimensional embedding of a high dimensional data set which retains any learnt manifold structure. That is, given a high-dimensional data set, a manifold learning algorithm will attempt to learn a lower dimensional manifold that the high dimensional data lies on or near. Many algorithms exist to perform this task, some of which aim to retain linear structure between the high and low-dimensional spaces and some of which attempt to learn and retain the intrinsic non-linear manifold. These latter non-linear algorithms will often attempt to learn the manifold by explicitly modelling the local or global properties of the manifold. Some of these algorithms are explained in more detail in Chapter 2. In this thesis we are interested in retaining the local properties of the manifold across a global scale by modelling the manifold as a set of small linear patches with are then re-created in the low-dimensional space. The basic idea of our piecewise-linear approach to manifold

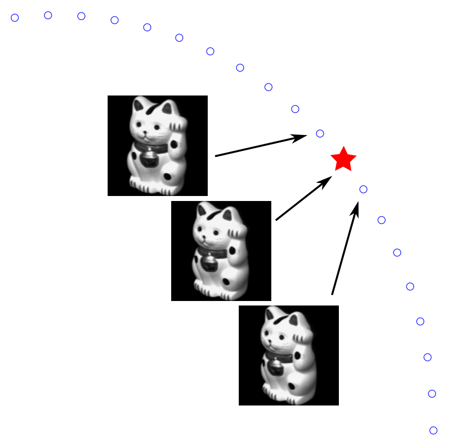


Figure 1.2: An example of embedding a new sample into a previously learnt low-dimensional embedding. The red star had not been learnt at the same time as the blue dots but a good out-of-sample extension algorithm should be able to successfully find its position on the low-dimensional manifold.

learning is to break the manifold up into small pieces that are locally linear and so locally low-dimensional. We then determine how these pieces are connected, that is, given one of these local pieces we know which other pieces are connected to it. This enables us to merge these pieces into one large piece by re-connecting them in a piecewise fashion. This gives us our low-dimensional approximation of the manifold that is both locally and globally stable.

Another related problem to manifold learning is that of the out-of-sample extension problem. The problem can be summarised as follows: given a set of high-dimensional data and its low-dimensional embedding, we wish to find the low-dimensional position of a novel data point that is not in the previously learnt data set, without re-learning the entire dataset (Figure 1.2). The position of this new data point could be computed by appending it to the previously learnt high-dimensional data and re-learning the entire dataset. However, this is computationally inefficient for large data sets and we wish to find a more elegant solution to the problem. The method we develop in this thesis to perform the out-of-sample extension task works by learning the local geometric transform that occurs in the new sample's neighbourhood within the previously learnt data as a result of manifold learning. By applying this geometric transform to the new sample we are able to successfully embed it into the previously learnt low-dimensional embedding.

1.3 Contributions

The main contributions to this thesis come in the form of publications and patents and are listed below:

- Harry Strange and Reyer Zwiggelaar, Classification Performance related to Intrinsic Dimensionality in Mammographic Image Analysis, Proceedings of the Thirteenth Annual Conference on Medical Image Understanding and Analysis, 2009, pp.219-223

- Harry Strange and Reyer Zwiggelaar, Iterative Hyperplane Merging: A Framework for Manifold Learning. In Frédéric Labrosse, Reyer Zwiggelaar, Yonghuai Liu, and Bernie Tiddeman, editors, Proceedings of the British Machine Vision Conference, pages 18.1-18.11. BMVA Press, September 2010

- Reyer Zwiggelaar and Harry Strange, Identification of Texture Connectivity, Patent KS.P47544GB, 2010

- Harry Strange and Reyer Zwiggelaar, Parallel Projections for Manifold Learning. In Proceedings of the Ninth International Conference on Machine Learning and Applications. Washington DC, December 2010. IEEE

Press.

- Harry Strange and Reyer Zwiggelaar, A Generalized Solution to the Out-of-Sample Extension Problem in Manifold Learning. In Proceedings of AAAI2011 the Twenty-Fifth Conference on Artificial Intelligence. San Francisco, CA, August 2011.

Second to these publications comes the impact of this thesis to the wider research community. We have demonstrated that the modelling of a manifold in a piecewise-linear fashion has many benefits and it is expected that more work will be carried out to overcome some of the current limitations to this approach that we have encountered.

1.4 Structure of Thesis

The rest of this thesis is structured as follows. We begin in Chapter 2 by introducing and exploring the area of manifold learning. Its history is traced from its earliest inception to present day state of the art techniques. The main techniques are described and phrased within a consistent mathematical algorithm. We also describe the core mathematics needed to work with manifold learning as well as current open-problems related to the field. In Chapter 3 we introduce the Piecewise-Linear Manifold Learning algorithm. This algorithm is described in both a high-level and low-level fashion, enabling the reader to either quickly gain insight into its functionality, or to delve deeper into its inner workings. Each step of the algorithm is explained in detail and various considerations and mathematical assumptions are discussed. Chapter 4 provides an in depth analysis into the performance of the Piecewise Linear Manifold Learning algorithm on both artificial and image data sets. The performance is measured using various error measures, as well as visualisations, and the algorithm is compared against the leading state of the art manifold learning techniques. Chapter 5 introduces the Generalised Out-of-Sample Extension method used to embed new samples into previously learnt embeddings. The results show its effectiveness and is compared against other out-of-sample techniques. Finally in Chapter 6 we draw the main conclusions from the work contained in this thesis and show how the aims we have set out to achieve have been met. We also point to possible future directions of research that have arisen as a result of the work contained in this thesis.

"Simplicity is removing the unnecessary so that the necessary may speak"

Hans Hofmann (1880 - 1966)

2 Manifold Learning

This chapter is concerned with providing the motivation for manifold learning as well as discussing the key contributions to the field. We begin in Section 2.1 by describing the practical and theoretical motivations for manifold learning. In Section 2.2 we outline the core mathematical framework within which we can introduce, expand and explain the mathematical principles for manifold learning. A history of manifold learning is then presented in Section 2.3 from the earliest day to the current state of the art. This section provides an overview of the evolution of strategies used to solve the problem of manifold learning and places them within a historical framework. We present a brief taxonomy of manifold learning algorithms in Section 2.4. In Section 2.5 we discuss the current open problems that researchers in the field face. Some places for further reading are outlined in Section 2.6 and we finish in Section 2.7 by drawing conclusions from this chapter and discussing how this guided our research in the following chapters.

2.1 Dimensions, Curses and Blessings

The field of manifold learning stems from the need to make sense of datasets with large numbers of samples and variables (i.e. high-dimensional data). Researchers in many diverse areas such as biology, engineering, medicine and remote sensing are presented with datasets with large numbers of observations and large numbers of variables taken for each observation. Advances in data collection devices has meant that often more variables are measured than are actually needed. This redundancy means that often the important variables, or dimensions, are ‘hidden’ within the data set. Traditional statistical analysis tools either fail to handle data sets with large numbers of dimensions or significantly degrade in performance. As such, it is useful to have a tool that can extract these hidden dimensions and remove the redundant ones. As well as improving the performance of statistical tools, this reduction of the number of dimensions will help with visualising the data. Humans struggle to visualise high-dimensional spaces. Anything where the dimensionality of the space is greater than three will be difficult to visualise. As such reducing the number of dimensions to a visualisable number will enable researchers to observe the data within a known frame of reference and therefore make it easier for them to attempt to draw conclusions as to the properties of the high-dimensional data.

A common term used to describe the problems associated with high-dimensional data is “the curse of dimensionality” (Bellman, 1961). This term underlies the theoretical reasons why working in high-dimensional spaces is problematic. Bellman considered a Cartesian grid of spacing 0.1 on the 10-dimensional unit cube. In this case the number of points on the cube equals 10^{10}. For the same scenario but considering a 20-dimensional unit cube the number of points increases to 10^{20}. Bellman’s interpretation of this is that the number of data points required to get a low variance estimate of a function of several variables grows exponentially with the number of dimensions. Put simply, as the dimensionality of the space increases so does the sparsity of the data. This "empty space phenomenon" (Scott and Thompson, 1983) gives unexpected properties to high-dimensional spaces (a matter discussed further in Section 2.5.5).

Traditionally two different approaches can be used to overcome these problems of dimensionality: feature selection and feature transformation. Both approaches attempt to build a lower-dimensional representation of high-dimensional data, however there are key differences in the way that each approach achieves this. Feature selection selects an optimal subset of the dimensions (features) that match some optimality criteria such as Minimum Description Length or Bayesian Information Criterion (Guyon and Elisseeff, 2003). Feature transformation transforms the high-dimensional data by projecting it into a learnt low dimensional space. Manifold learning and dimensionality reduction algorithms fall into the feature transformation category. These algorithms work by transforming the original features into a new low-dimensional space. As such it is to feature transformation that we will be turning our attention throughout the remainder of this chapter and the remainder of this thesis.

2.2 Mathematical Preliminaries

To be able to succinctly describe the mathematical processes that drive manifold learning techniques we need to establish a common mathematical framework within which we can work. To do this we need to define concepts from various mathematical disciplines and so this section sets out the mathematical terminology that will be used in the remainder of this thesis.

2.2.1 Vector and Matrices

A vector is an ordered collection of n real numbers denoted

$$v = [v_1, v_2, \dots, v_n] \tag{2.1}$$

where each of these numbers are referred to as a component of v. We refer to this vector as an n-dimensional vector. There are two ways of representing such a vector, either as a row-vector or as a column vector. A row vector will take the form as shown in Equation 2.1 where as a column vector takes the form

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \tag{2.2}$$

Unless otherwise stated, whenever we refer to a vector we will assume that they take the form of a row vector.

A matrix is a rectangular array of nm numbers with n-rows and m-columns such that

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} \tag{2.3}$$

Throughout this thesis matrices are represented in bold type. The above matrix has dimensions of m by n and each number (e.g. a_{ij}) is referred to as an element of $\mathbf{A}$.

2.2.2 Eigenanalysis

The term eigenanalysis is derived from the German word *eigen* meaning "own" or "characteristic". Eigenalysis is concerned with the study of eigenvectors and eigenvalues related to a square matrix, $\mathbf{A}$. An eigenvalue of $\mathbf{A}$ is a scalar which we denote as λ. An eigenvector of $\mathbf{A}$ is a *non-zero* column vector denoted as v. For a given square matrix, $\mathbf{A}$, all eigenvalues and eigenvectors must satisfy the equation

$$\mathbf{A}v = \lambda v \tag{2.4}$$

and we say that λ is the corresponding eigenvalue to the eigenvector v. The eigenvalues can be calculated by rewriting Equation 2.4 as

$$(\mathbf{A} - \lambda \mathbf{I})v = 0 \tag{2.5}$$

where $\mathbf{I}$ is an $n \times n$ identity matrix. A non-trivial solution to this linear system exists if the determinant equals zeros[1], that is

$$\det(\mathbf{A} - \lambda \mathbf{I}) = 0 \tag{2.6}$$

This equation is known as the Characteristic Equation, or Characteristic Polynomial, (Weisstein, 2011). The roots of this polynomial gives us the eigenvectors

[1]This arises from Cramer's rule which provides a solution of a system of linear equations where there are as many equations as unknowns.

of $\mathbf{A}$.

Once the eigenvalues have been calculated as above then the eigenvectors can be found by using the linear system in Equation 2.5 and substituting the eigenvalues found from the roots of the characteristic polynomial to calculate each corresponding eigenvector.

Eigendecomposition therefore is the factorisation of a square $n \times n$ matrix into its associated eigenvalues and eigenvectors. Given an $n \times n$ matrix, $\mathbf{A}$, it can be factorised as

$$\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^{-1} \tag{2.7}$$

where $\mathbf{V}$ is a matrix containing as columns the n eigenvectors such that v_i corresponds to the i^{th} column of $\mathbf{V}$ and therefore the i^{th} eigenvector of $\mathbf{A}$. Λ is a diagonal matrix with the eigenvalues of $\mathbf{A}$ residing along the diagonal of Λ. That is, $\lambda_i = \Lambda_{ii}$. It is also worth noting that $\mathbf{A}$ must be a diagonalisable matrix, that is there exists an invertible matrix $\mathbf{P}$ such that $\mathbf{P}^{-1}\mathbf{A}\mathbf{P}$ is a diagonal matrix.

2.2.3 Vector Spaces

A vector space is formed by a set of vectors, V, on which certain additive and multiplicative properties hold. We denote a vector, v, as belonging to a vector space, V, as $v \in V$. As shown by Blyth and Robertson (2007), for V to be a vector space the following properties must hold:

1. $x + y = y + x$ for all $x, y \in V$;

2. $(x + y) + z = x + (y + z)$ for all $x, y, z \in V$;

3. there exists an element $0 \in V$ such that $x + 0 = x$ for every $x \in V$;

4. for every $x \in V$ there exists $-x \in V$ such that $x + (-x) = 0$;

5. $\lambda(x + y) = \lambda x + \lambda y$ for all $x, y \in V$ and all scalars λ;

6. $(\lambda + \mu)x = \lambda x + \mu x$ for all $x \in V$ and all scalars λ, μ;

7. $(\lambda\mu)x = \lambda(\mu x)$ for all $x \in V$ and all scalars λ, μ;

8. $1x = x$ for all $x \in V$.

When the above scalars are real numbers we say that V is a real vector space[2].

Throughout this thesis when dealing with vector spaces we deal with Euclidean vector spaces. The Euclidean vector space is the set of n real numbers, $\mathbb{R}^n$, with an added Euclidean metric such that distances between two points, $x, y \in \mathbb{R}^n$, are defined as

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.8}$$

As well as the above Euclidean metric, the Euclidean vector space has an inner product between two vectors, $x, y \in \mathbb{R}^n$, defined as

$$x \cdot y = x_1 y_1 + x_2 y_2 + \ldots + x_n y_n \tag{2.9}$$

2.2.4 Transformations

Given two vector spaces V and W, a linear transformation T from V into W is a function that assigns each vector $\mathbf{v} \in V$ a unique vector $T\mathbf{v} \in W$ such that, for each $\mathbf{u}$ and $\mathbf{v}$ in V and each scalar α

[2]Similarly, if the scalars are complex numbers we refer to V as a complex vector space

$$T(\mathbf{u}+\mathbf{v}) = T\mathbf{u} + T\mathbf{v} \tag{2.10}$$

and

$$T(\alpha\mathbf{v}) = \alpha T\mathbf{v} \tag{2.11}$$

This transformation is denoted $T : V \to W$ which indicates that T takes the real vector space V into the real vector space W, and T is a function with V as its domain and a subset of W as its range (Grossman, 1994). This can be phrased in terms of a transformation matrix such that for a linear transformation $T : \mathbb{R}^n \to \mathbb{R}^m$ there exists a unique $m \times n$ transformation matrix $\mathbf{A}_T$ such that

$$T\mathbf{x} = \mathbf{A}_T\mathbf{x} \tag{2.12}$$

That is the transformed image, $T\mathbf{x}$, is a product of $\mathbf{x}$ and the transformation matrix $\mathbf{A}_T$.

2.2.5 From Subspaces to Manifolds

The above transformations only deal with the global linear case. A linear transform $T : \mathbb{R}^p \to \mathbb{R}^q$ will map onto a global linear subspace in $\mathbb{R}^q$. However, data is often not distributed on a single linear subspace. Rather, there is often curvature and non-linear variation within the data and as such a linear transform will not adequately capture the true characteristics of the data. This is where the concept of manifolds become important.

Manifolds extend the idea of surfaces to higher-dimensions. Manifolds are often globally non-linear yet locally exhibit linear structure (Roweis and Saul, 2000). Concepts from differential geometry show that the local metric on a small local neighbourhood of a manifold, combined with the topology information of these neighbourhoods, fully determine a manifold's intrinsic geometry (Carmo, 1992; Brand, 2004). As such, by modelling data using the manifold structure one can often extract the true data distribution that is not available when modelling the data using subspace methods.

A manifold $\mathcal{P}$ can be thought of as locally isometric to an open subset of $\mathbb{R}^q$ and embedded in high-dimensional Euclidean space $\mathbb{R}^p$ (where $q \ll p$). As such, at a local scale $\mathcal{P}$ is homeomorphic to Euclidean space and for an infinitesimal neighbourhood $\mathcal{N} \subset \mathcal{P}$ we can say that $\mathcal{N} \subset \mathbb{R}^q \in \mathbb{R}^p$.

Given a manifold in high-dimensional space, $\mathcal{P} \in \mathbb{R}^p$, that is locally isometric to $\mathbb{R}^q$ then we denote the embedding of this manifold in $\mathbb{R}^q$ as $\mathcal{Q}$. We represent this function as $f : \mathcal{P} \hookrightarrow \mathcal{Q}$ where the hooked arrow, $\hookrightarrow$, represents the embedding operator.

2.3 A History of Manifold Learning

Now that we have laid the foundations of manifold learning in terms of practical motivation and mathematical preliminaries, we can turn our focus to the history of manifold learning.

The story of manifold learning can be broadly split into two main time periods. First the establishment of dimensionality reduction and subspace learning which spanned the first 100 years of research in the area. This period was vital in the laying down of foundations which would later prove essential for research into manifold learning. The second period was triggered by the establishment of the

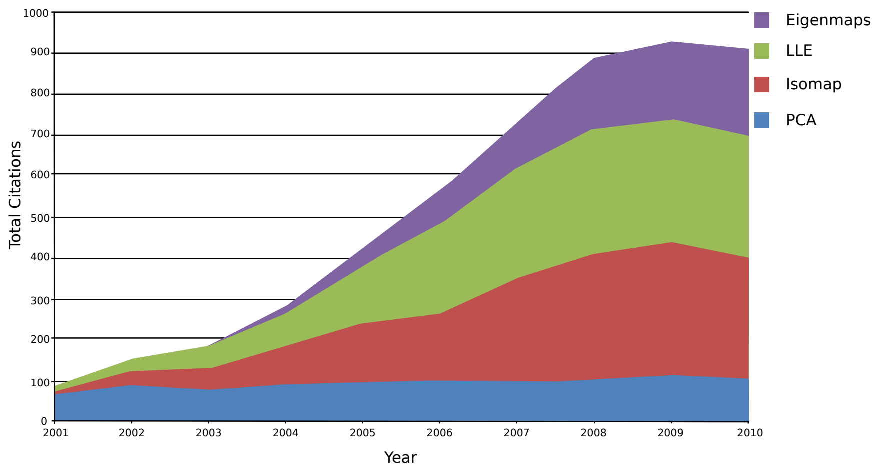


Figure 2.1: The total citations from 2001-2010 of the three most popular manifold learning algorithms: LLE (Roweis and Saul, 2000), Isomap (Tenenbaum et al., 2000) and Laplacian Eigenmaps (Belkin and Niyogi, 2002) as well as PCA (Wold et al., 1987).

first true manifold learning algorithms. This period of time is a factor of ten shorter than the first but has seen a rapid growth in the number of publications related to the field of manifold learning.

Figure 2.1 shows the relative contributions of LLE (Roweis and Saul, 2000), Isomap (Tenenbaum et al., 2000) and Laplacian Eigenmaps (Belkin and Niyogi, 2002) as well as PCA (Wold et al., 1987) over the time period 2001 to 2010. The citation data was taken from the ISI Web of Knowledge database[3]. The PCA citation data was taken from the most recent, and most cited, review paper on PCA (Wold et al., 1987) as the Web of Knowledge database does not go back far enough to contain citation data for the original PCA papers. As can be seen, the manifold learning algorithms very quickly overtake PCA in terms of percentage contribution of citations although PCA remains relatively constant throughout the time period. This shows that not only has manifold learning become a lucrative research area, but also that PCA remains an important subject that

[3]`http://wok.mimas.ac.uk` (Link checked 27/07/2011).

has not been ‘overtaken’ as a result of the advent of non-linear dimensionality reduction techniques.

To understand the current state of manifold learning we first need to explore the prior work that underpins many of the recent manifold learning algorithms. So it is to this first 100 years that we initially turn our attention.

2.3.1 The First 100 Years: 1900-2000

Principal Components Analysis (1901)

Arguably one of the most important discoveries in the field of statistical analysis in the 20th century was the discovery of Principal Components Analysis (PCA) by Pearson (1901). The algorithm was later re-discovered by Hotelling (1933), Karhunen (1946) and Loéve (1948) and it provides the backbone to research in the field of dimensionality reduction and manifold learning. The simplicity and effectiveness of PCA means that it has been widely used in a range of application areas and is still the *goto* technique for many researchers over 100 years after its discovery.

The basic premise of PCA is to find the low-dimensional subspace contained within the high-dimensional data which holds most of the ‘information’. By projecting the high-dimensional samples onto this lower-dimensional subspace the important information is retained while the less important information (i.e. noise) is removed. Since PCA underpins so many dimensionality reduction and manifold learning algorithms it is worth spending some time unpacking the algorithm and understanding the underlying mathematics.

As with all dimensionality reduction and manifold learning techniques, PCA takes as input a high-dimensional matrix $\mathbf{X} \in \mathbb{R}^p$, of row vectors, and produces a low-dimensional representation $\mathbf{Y} \in \mathbb{R}^q$ and $q < p$. To produce this matrix, $\mathbf{Y}$, PCA finds the low-dimensional subspace within $\mathbf{X}$ that contains the most variance. This maximum variance subspace equates to the subspace which holds most of the information. As shown in (Lee and Verleysen, 2007), PCA can be derived from several criteria. Pearson (Pearson, 1901) derives the solution to PCA from the minimal reconstruction error criteria, whereas Hotelling (1933) uses the maximal preserved variance criteria[4]. Most manifold learning algorithms derive their data model from Hotelling's criteria.

Hotelling's PCA can be split up into 3 main steps, 1) producing a matrix that describes the variance of each sample to each other sample, 2) performing eigendecomposition of this matrix to produce a set of associated eigenvalues and eigenvectors, 3) projecting the original data onto the top q eigenvectors (sorted according to their eigenvalues) to produce the low-dimensional representation. These three steps are still followed in many manifold learning algorithms and can be generalised as follows:

1. **Feature Step:** Build a square symmetric feature matrix from the original input data
2. **Decomposition Step:** Decompose this matrix into associated eigenvalues and eigenvectors
3. **Projection Step:** Project the original data onto these eigenvectors ordered according to their associated eigenvalues

As will be seen later, the final projection step is often discarded, but the process of building a descriptive feature matrix of the original data and then using eigendecomposition to find its associated eigenvectors is still the backbone of most manifold learning algorithms.

[4]A third criterion - distance preservation - is shown when discussing Multidimensional Scaling.

At the centre of PCA is the model that assumes that the low-dimensional embedding is the result of a linear transformation $\mathbf{W}$ such that

$$\mathbf{Y} = \mathbf{X}\mathbf{W} \tag{2.13}$$

This transformation $\mathbf{W}$ projects the data between two co-ordinate spaces such that $\mathbf{W} : \mathbb{R}^p \to \mathbb{R}^q$. The central question then becomes how does PCA form this transformation matrix? As discussed previously we can build this transformation matrix by following the steps outlined above.

The first step is to build the symmetric feature matrix which, according to Hotelling's methodology, is the covariance matrix. This matrix describes the variance between a sample and every other sample in the data. To build the covariance matrix the samples need to be centred such that, as far as possible, $\bar{\mathbf{X}} = 0$ (i.e. the samples in $\mathbf{X}$ are zero mean). The covariance matrix can then be defined as

$$\mathbf{C}_{ij} = E\{(x_i - E(x_i))(x_j - E(x_j))\} \tag{2.14}$$

where E is the expected value. Once we have found the covariance matrix it is decomposed to its eigenvectors and eigenvalues to obtain the basis vectors of the subspace spanned by the covariance of the data. This subspace describes the directions of variance, or principal directions, of the data. The covariance matrix can be factored by eigendecomposition:

$$\mathbf{V}\Lambda = \mathbf{C}\mathbf{V} \tag{2.15}$$

where Λ is a square matrix containing the eigenvalues along its diagonal. $\mathbf{V}$ contains as columns the eigenvectors of the matrix $\mathbf{C}$. The i-th eigenvalue in Λ relates to the 'size' of the i-th eigenvector. As such, the eigenvectors are ordered according to the descending ordering of the eigenvalues. This now gives rise to the basis vectors for the variance subspace spanning $\mathbf{X}$. To find $\mathbf{Y}$ the top q eigenvectors of $\mathbf{V}$ are taken and project $\mathbf{X}$ onto them such that

$$\mathbf{Y} = \mathbf{X}\mathbf{V}_{1 \ldots q} \tag{2.16}$$

This gives the low-dimensional representation, $\mathbf{Y}$, which has been found by projecting $\mathbf{X}$ onto the subspace of maximum variance.

PCA has been used in many different fields (e.g. (Turk and Pentland, 1991; Huber et al., 2005)) and has also been heavily modified and extended since its original definition. Simple PCA (Partridge and Calvo, 1998) is an iterative method for calculating the principal components that does not require the eigendecomposition of a covariance matrix. Rather it works by iteratively finding the principal components of the data set and as such is closely related to Hebbian learning (Diamantaras and Kung, 1996). One of the advantages of using Simple PCA is that it overcomes the problem that PCA encounters with very high-dimensional datasets. Since the size of the covariance matrix is proportional to the dimensionality of the high-dimensional space, with very high-dimensional data the calculation and eigendecomposition of this matrix will become computationally unfeasible. As such iterative methods such as Simple PCA or Probabilistic PCA (Roweis, 1997) may be used.

As previously discussed, PCA can be formulated using three criteria. The third criteria is to formulate it using distances which gives rise to one of the main extensions of PCA: Classical Multidimensional Scaling (Torgerson, 1952). Multidimensional Scaling (MDS) works in the same way as PCA but replaces the covariance matrix with a Gram matrix that represents the inner product between each data point. This then gives us a distance based adaptation of PCA[5].

Sammon Mapping (1969)

Since PCA was first discovered in 1901 it was rediscovered and reformulated many times (e.g. Karhunen (1946); Loéve (1948)) most notably in the form of Classical MDS (Torgerson, 1952). However, it was not until 1969 that an alternative approach to dimensionality reduction was suggested: Sammon Mappings (Sammon, 1969). Sammon Mappings extend the ideas of MDS so as to attempt to preserve non-linear, rather than just linear, relationships in the data. This is done by reconstructing the interpoint distances in the low-dimensional space according to the distances measured in the high-dimensional space. Sammon proposed the mapping error function ϵ which can be minimised using a steepest descent algorithm as

$$\epsilon = \frac{1}{\sum_{i<j} d^*_{ij}} \sum_{i<j}^{n} \frac{(d^*_{ij} - d_{ij})^2}{d^*_{ij}} \tag{2.17}$$

This function seeks to minimise the distance between two points as measured in the high-dimensional space, d^*_{ij}, and the same points in the low-dimensional space, d_{ij}. Prior to the minimisation of ϵ the vectors in $\mathbf{Y}$ are assigned initial approximations based on PCA.

Sammon Mappings represented an important step forward in the history of manifold learning. It signified the start of research outside of the bounds of

[5]The connection between PCA and MDS is described in more detail in (Williams, 2002; Platt, 2005)

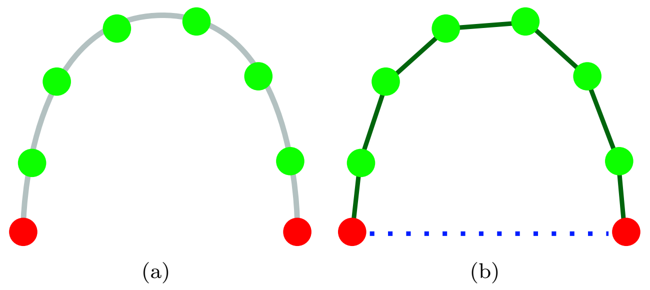


Figure 2.2: (a) shows points sampled from a simple curved surface. (b) shows the two different types of distances, the dotted blue line represents the Euclidean distance between the two red points. The solid green line in (b) shows the geodesic distance between the two red points. The geodesic distance is closer to the true manifold distance between the points.

the linear assumption, although the algorithm's ability to successfully embed non-linear manifolds is questionable. The core of the algorithm is to re-build in the low-dimensional space the distances calculated in the high-dimensional space. However, measuring global distances that preserve manifold relations is problematic. The distance relations between points can be accurately modelled at a local scale using the Euclidean distance but not so at a global scale. Figure 2.2 illustrates this problem. So although Sammon Mappings can perform well on some non-linear data, on most it will show little improvement over PCA and Classical MDS. As such, Sammon Mappings is a half-way house between linear and non-linear techniques. Despite this, the algorithm is still an important milestone in the history of manifold learning, as it would be almost three decades until a truly viable, and truly non-linear, method was developed.

Kernel PCA (1998)

Throughout the 1980s and early 1990s methods were proposed to perform dimensionality reduction based on artificial neural networks[6]. These approaches presented a very different paradigm to dimensionality reduction. Rather than

[6]See Section 2.3.3 for a discussion as to why these methods are omitted from this discussion.

drawing inspiration from statistical or geometric sources, they tackled the problem using methods from artificial intelligence and neural networks. These methods were some of the first to present possible solutions to the non-linear dimensionality reduction problem. It was not until 1998 that a method akin to the earlier dimensionality reduction techniques was presented to tackle the non-linear dimensionality reduction problem.

Kernel PCA (Scholkopf et al., 1998) is a non-linear form of traditional PCA that exploits the 'kernel trick' (Shawe-Taylor and Christianini, 2004) to reformulate the traditional linear problem into that of a high-dimensional kernel space problem. The core of Kernel PCA is to move the problem from that of finding the principal components of the input space vectors to that of finding the principal components of the variables that are non-linearly related to the input variables (Scholkopf et al., 1998).

Given a dataset, $\mathbf{X}$, a positive semi-definite kernel, κ, is defined as a real value function, Φ, which maps the points into a dot product space $\mathcal{H}$ such that, $\Phi : \mathbf{X} \rightarrow \mathcal{H}$. Kernel PCA therefore computes the principal components of the points $\{\Phi(x_1), \Phi(x_2), \ldots, \Phi(x_n)\}$. The kernel matrix, $\mathbf{K}$, is created with elements such that

$$k_{ij} = \kappa(x_i, x_j) \tag{2.18}$$

$\mathbf{K}$ is then double centred (a step similar to subtracting the mean of the features in traditional PCA) leading to the data being zero-mean in kernel space. The top q eigenvectors, $\{v_1, v_2, \ldots, v_q\}$ of $\mathbf{K}$ are then computed. The eigenvectors of the covariance matrix in the kernel space can be constructed and are related to the eigenvectors of $\mathbf{K}$ through

$$a_i = \frac{1}{\sqrt{\Lambda_i}} v_i \tag{2.19}$$

To obtain a low-dimensional embedding the data is projected onto the eigenvectors of the covariance matrix, a_i. The low-dimensional representation is then given by

$$y_i = \left\{ \sum_{j=1}^{n} a_1^j \kappa(x_j, x_i), \dots, \sum_{j=1}^{n} a_q^j \kappa(x_j, x_i) \right\} \tag{2.20}$$

where a_1^k is the j-th value of the vector a_1. Here κ is the same kernel function used to compute the kernel matrix.

The performance of Kernel PCA is heavily reliant on the choice of kernel function used. If using a linear kernel then Kernel PCA becomes traditional PCA. Other possible kernels include the polynomial kernel and the Gaussian kernel.

Kernel PCA has been applied to areas such as face recognition (Kim et al., 2002) and speech recognition (Lima et al., 2004). However, the size of the kernel matrix is proportional to the square of the number of data points. This makes Kernel PCA unusable in applications with large datasets. An approach to overcome this was proposed in (Tipping, 2000).

One interesting extension of Kernel PCA is to phrase existing manifold learning algorithms within the Kernel PCA framework (Ham et al., 2003). Isomap, Laplacian Eigenmaps and Locally Linear Embeddings are all described as Kernel PCA with specially constructed Gram matrices. This idea was further extended when considering the out-of-sample extension in (Bengio et al., 2003).

Although the theoretical framework for Kernel PCA is both simple and elegant it is not suitable for embedding purposes (Lee and Verleysen, 2007). Kernel PCA has no geometric interpretation that aids users in the choice of kernel functions. Rather, extensions of Kernel PCA such as (Ham et al., 2003; Bengio et al., 2003) or Maximum Variance Unfolding (Weinberger and Saul, 2006b) should be used.

2.3.2 The Manifold Revolution: 2000-Present Day

Although some non-linear techniques were developed prior to the year 2000, very little was discussed in terms of manifold learning. The existing non-linear techniques, such as Kernel PCA or Sammon Mappings, sought to find non-linear mappings but did not explicitly attempt to reconstruct the manifold upon which the data is thought to lie. As such there is a clear distinction between those techniques which seek to perform non-linear dimensionality reduction (such as Kernel PCA and Sammon Mappings) and those which seek to perform manifold learning. Even though manifold learning can be seen as a form of non-linear dimensionality reduction, it is still worth making the distinction so as to be clear as to which techniques explicitly learn the low-dimensional manifold and those which do not.

In this section we present the algorithms that we believe have contributed the most to the field of manifold learning since the year 2000. These techniques either provide real improvements over previous techniques or represent a completely different approach to the manifold learning problem. At the end of this section we briefly discuss other manifold learning algorithms that have emerged during this period.

Isomap (2000)

Isomap was first discussed in 1998 (Tenenbaum, 1998) but was most famously presented alongside its main rival algorithm, Locally Linear Embedding, in Science in 2000 (Tenenbaum et al., 2000). These two algorithms were presented side by side in the same issue and present two very different approaches to manifold learning.

Isomap explicitly builds on the previous strengths of PCA and MDS: computational efficiency; global optimality; and asymptotic convergence guarantees; but extends them to learn a broader class of non-linear manifolds. Central to the Isomap algorithm is the geodesic distance between samples. In PCA and MDS the relationships between points are measured at a local and global scale using measures such as covariance or Euclidean distance. However, for non-linear manifolds these relationships fail to adequately model the underlying manifold (See Figure 2.2). Isomap seeks to overcome this problem by measuring inter-point distances *along the manifold* by using geodesic distances. The simple and elegant solution that Isomap employs to measure geodesic distances is partly what gives it its attractiveness. For neighbouring points the Euclidean distance provides a good approximation to the geodesic distance. For far away points the geodesic distance can be approximated by summing a sequence of 'short hops' between neighbouring points. Bernstein et al. (2000) showed that these approximations can be computed by finding shortest paths in a graph with the vertex set of the graph being set to the input data points and the edge set connecting neighbouring samples. As such Isomap was not just one of the first true manifold learning algorithms but also the first in the class of graph based (spectral) manifold learning algorithms.

The two core steps of the Isomap algorithm are the construction of a graph between samples and the calculation of geodesic distances. As stated in the original Isomap paper the neighbourhood graph can be constructed in one of two ways. Both seek to create a graph $G = \langle V, E \rangle$ where V is the vertex set which is equal to the samples in $\mathbf{X}$, and the edge set, E, contains the description of connectivity between vertices at a local scale. The edges can either be set according to the k-nearest neighbour rule or the ϵ-neighbourhood rule. The k-nearest neighbour rule connects a point to its k-nearest points ordered according to the inter-point Euclidean distance. The ϵ-neighbourhood rule connects all points that are within a ball of radius ϵ around the given data point.

The neighbourhood graph can now be fully defined as $G = \langle V, E \rangle$ with $V = \{x\}_{i=1}^{n}$ and

$$E_{ij} = \begin{cases} \parallel x_i - x_j \parallel & \text{if} \qquad x_j \in \mathcal{N}_i \\ \infty & \text{otherwise} \end{cases} \tag{2.21}$$

where $\mathcal{N}_i$ is the set containing the nearest neighbours of x_i according to the k-nearest neighbour rule or the ϵ-neighbourhood rule.

This graph now describes the local connectivity of the data. To obtain global connectivity information the geodesic distances are calculated using a shortest path algorithm between all pairs of vertices (e.g. Dijkstra's algorithm (Dijkstra, 1959)). This graph of shortest paths is described by a square symmetric matrix, $\mathbf{D}$, such that $d_{ij} = d_{ji}$, and d_{ij} is equal to the geodesic distance between sample x_i and sample x_j. Phrasing this problem in terms of PCA or MDS, this distance matrix can now be thought of as the Isomap equivalent to the covariance matrix (PCA) or the standard Euclidean distance matrix (MDS).

The final step of the Isomap algorithm is to construct the low-dimensional embedding, and as with PCA this is done by solving the eigenproblem

$$\Lambda \mathbf{V} = \mathbf{D}\mathbf{V} \tag{2.22}$$

However, since Isomap is a non-linear mapping the original data cannot be simply projected onto the eigenvectors $\mathbf{V}$. This means that Isomap does not follow the third step of the three steps described earlier. Rather it uses the methodology found in MDS rather than PCA, so the low-dimensional vector y_i is calculated

$$y_i = [\sqrt{\lambda_1} v_1^i, \sqrt{\lambda_2} v_2^i, \ldots, \sqrt{\lambda_q} v_q^i] \tag{2.23}$$

where λ_q is the q-th ordered eigenvalue and v_q^i is its associated eigenvector.

As mentioned earlier, Isomap is able to learn a broader class of non-linear manifolds than PCA and MDS. However, Isomap is not without its short comings. The topological stability of Isomap was brought into question by Balasubramanian and Schwartz (2002). This problem of topological stability stems from the fact that Isomap will obtain a globally consistent mapping of the manifold without considering the changes that occur in local topology as a result of applying this mapping. Of more importance however are the problems of the correct parameter choice and Isomap's large computational complexity. The question of what neighbourhood size parameter should be selected is still an open problem and is discussed in more detail in Section 2.5.2, but for now it is worth noting that incorrect parameter selection can lead to one of two failure cases. First, if the choice of k or ϵ is too small then a disconnected manifold is produced. As such, Isomap will fail to recover the true manifold as there is no global information contained in the graph. Rather, the manifold is split into smaller disconnected sub-manifolds[7]. Secondly, if the value of k or ϵ is too large then erroneous connections will be made and the neighbourhood graph will 'short circuit' the manifold. This will lead to incorrect geodesics beings calculated as edges will exist in E that do not represent the true connectivity of the manifold.

The computational bottlenecks in the Isomap algorithm are the calculation of all shortest paths and the eigendecomposition of the resulting shortest path matrix. Landmark Isomap (Silva and Tenenbaum, 2003a) seeks to overcome these problems by using landmark points drawn from the input data rather than the entire input data. Once the landmark points have been embedded the remaining points can be located by using the known distances from the landmark points as constraints. This considerably speeds up the algorithm if the number of landmark points is less than the number of input points. However, if too few landmark points are chosen then Landmark Isomap will be unable to recover the low-dimensional embedding as the input data has become too sparse.

Since its introduction in 2000 there have been many proposed extensions and variants to the Isomap algorithm (e.g. Lee et al. (2004); Yang (2004); Brun et al. (2005)). One variant of real interest is Conformal Isomap (C-Isomap) (Silva and Tenenbaum, 2003a), which seeks to improve upon normal Isomap by adding weights to each edge of the neighbourhood graph that are based on the mean distance between each data point and its k-nearest neighbours. Although C-Isomap provides an improvement over normal Isomap in embedding quality it increases its complexity as a larger sample size is required (this is due to the fact that the embedding depends on two approximations - geodesic distance and data density - rather than just one).

[7]One way to overcome this problem is to only embed the largest of these sub-manifolds.

Isomap represents a landmark in the history of manifold learning as it is arguably the first manifold learning algorithm. What is surprising is that it is still one of the most effective and powerful manifold learning algorithms even though new approaches have since been presented. Isomap remains the cornerstone of manifold learning research, and all signs point to it remaining so for some time.

Locally Linear Embedding (2000)

As mentioned earlier, Isomap was presented in an edition of the journal Science in 2000 alongside another very different manifold learning algorithm, Locally Linear Embedding (LLE) (Roweis and Saul, 2000). LLE presents a different paradigm to manifold learning than that of Isomap. Whereas Isomap aims to recover global properties of the manifold by measuring interpoint geodesic distances, LLE aims to preserve local properties of the data and to recover a global non-linear manifold structure by locally linear fits. The two different approaches taken by Isomap (global) and LLE (local) are still two of the main streams of research in manifold learning. Although more sophisticated and mathematically elegant methods have been proposed since their discovery they still remain the two benchmark manifold learning algorithms.

Although local methods had been previously discovered (e.g. Local PCA (Fukunaga and Olsen, 1971)) they had not attempted to produce a global embedding of the data. Rather, previous local methods embedded data points into their individual local coordinate space as opposed to a global coordinate space. LLE aims to preserve local properties of the data whilst also embedding the data points into a continuous global coordinate space.

The central assumption behind LLE is that each sample and its k-nearest neighbours lie on a locally linear patch of the manifold (Roweis and Saul, 2000). If this assumption holds then each sample can be reconstructed by linear coefficients of its neighbours such that the error measure of reconstructing x_i in terms of its neighbours is

$$\mathcal{E} = \sum_{i=1}^{n} \|x_i - \sum_{j=1}^{n} w_{ij} x_j\|^2 \tag{2.24}$$

The matrix $\mathbf{W}$ is a matrix of weights such that w_{ij} summarises the contribution that x_j makes to the reconstruction of x_i. Two constraints are added to this cost function prior to optimisation. First, the samples are reconstructed only in terms of their neighbours such that $w_{ij} = 0$ if x_j is *not* in the neighbourhood of x_i. Secondly the rows of the weight matrix $\mathbf{W}$ are set to sum to 1, that is $\sum_{j=1}^{n} w_{ij} = 1$. This means that LLE becomes a sparse spectral problem. The optimal weights for $\mathbf{W}$ can be found by solving a least squares problem. Considering a single data point, x_i, and its nearest neighbours, $\mathcal{N}_j$, with reconstruction weights, w_j, then the reconstruction error, $\| \; x_i - \sum_{j=1}^{k} w_j \mathcal{N}_j \; \|^2$ is minimised in three steps. First, the inner products between the neighbours in $\mathcal{N}_j$ are calculated to form the correlation matrix, $\mathbf{C}_j$. Second, the Lagrange multiplier, λ, that enforces the sum to one constraint is calculated such that

$$\lambda = \frac{1 - \sum_{jk} \mathbf{C}_{jl}^{-1}(x \cdot \mathcal{N}_k)}{\sum_{jk} \mathbf{C}_{jk}^{-1}} \tag{2.25}$$

Finally, the reconstruction weights are computed as follows:

$$w_j = \sum_k \mathbf{C}_{jk}^{-1}(x \cdot \mathcal{N}_k + \lambda) \tag{2.26}$$

One important property of the reconstruction weight matrix, $\mathbf{W}$, is that it characterises the intrinsic geometric information of each neighbourhood invariant to translations, rotations and scaling. The importance of this becomes apparent when you consider that the weights used to construct $\mathbf{X}$ in $\mathbb{R}^p$ can also be used to reconstruct $\mathbf{Y}$ in $\mathbb{R}^q$. This is the heart of LLE: since the weights have been found in Equation 2.24, we can minimise the same cost function but now in terms of the low-dimensional vectors to obtain an estimation of the low-dimensional data:

$$\varphi(\mathbf{Y}) = \sum_{i=1}^{n} \|y_i - \sum_{j=1}^{n} w_{ij} y_j\|^2 \tag{2.27}$$

Whereas in Equation 2.24 we fixed the input vectors and minimised the weight matrix, here we have fixed the weight matrix and are minimising the low-dimensional vectors. Roweis and Saul (2000) showed that since this problem is well posed, the optimisation of the function $\varphi(\mathbf{Y})$ can be performed by solving a sparse $n \times n$ eigendecomposition whose bottom q eigenvectors provide a set of orthogonal coordinates centred on the origin. By phrasing LLE as a sparse eigenproblem the need to solve a large dynamic programming problem is avoided.

The popularity of LLE has lead to it becoming one of the most widely researched manifold learning algorithms with many variants being proposed (e.g. Donoho and Grimes (2003); He et al. (2005); Yin et al. (2007)). One particularly popular extension is Stochastic Neighbour Embedding (SNE) (Hinton and Roweis, 2000). SNE uses a probabilistic framework to position the data points in the low-dimensional space. Asymmetric probabilities are assigned to the data points in high-dimensional space and these probabilities are then reconstructed in the low-dimensional space by minimising the Kullback-Leibler divergences between the distributions in the high and low-dimensional spaces. As such, SNE can be thought of as a probabilistic extension to LLE. Other variants of SNE include t-SNE (Maaten and Hinton, 2008) and Trust Region SNE (Nam et al., 2004).

Despite its popularity, LLE is not without its weaknesses. LLE has difficulties with manifolds containing holes (Roweis and Saul, 2000) and also has a tendency to 'collapse' large portions of the manifold in low-dimensional space. Having said this, LLE is arguably still one of the most popular manifold learning algorithms. It is still the standard local manifold learning algorithm, and like Isomap it remains one of the foundational manifold learning algorithms. Even though different approaches to local manifold learning have been presented LLE still remains the most popular and most elegant.

Laplacian Eigenmaps (2002)

Laplacian Eigenmaps (Belkin and Niyogi, 2002) is another local technique that seeks to preserve local properties of the manifold. As with Isomap and LLE a neighbourhood graph is constructed by connecting the k-nearest neighbours of each datapoint. The distances contained in this graph are then reconstructed in a weighted manner in the low-dimensional space such that the distance between a point and its first neighbour contributes more than the distance to the second neighbour and so on.

To begin, Laplacian Eigenmaps constructs a neighbourhood graph that connects each data point to its k-nearest neighbours. For each vertex in this graph a weight is added to a sparse adjacency matrix, $\mathbf{W}$. The weights are based on the Gaussian kernel function such that if there exists an edge between x_i and x_j the weight is

$$w_{ij} = e^{-\frac{\|x_i - x_j\|^2}{t}} \tag{2.28}$$

where t is the width of the kernel[8]. Laplacian Eigenmaps then seeks to minimise the following cost function

$$\varphi(\mathbf{Y}) = \sum_{ij} \parallel y_i - y_j \parallel^2 w_{ij} \tag{2.29}$$

One of the key aspects of the Laplacian Eigenmaps algorithm is that large values of $\mathbf{W}$ relate to small distances between points in the high-dimensional space. As such, nearby points in the high-dimensional space will be placed close together in the low-dimensional space and faraway points in the high-dimensional space will be placed further away in the low-dimensional space.

Equation 2.29 can be formulated as an eigenproblem by computing the degree matrix, $\mathbf{M}$, and the Laplacian, $\mathbf{L}$, of $\mathbf{W}$ (Anderson and Morley, 1985). The graph Laplacian $\mathbf{L}$ is found by $\mathbf{L} = \mathbf{M} - \mathbf{W}$ and the low-dimensional representation can be found by solving the eigenproblem

$$\mathbf{VL} = \Lambda\mathbf{MV} \tag{2.30}$$

The top q nonzero eigenvectors of $\mathbf{V}$, ordered according to their eigenvalues, Λ, correspond to the low-dimensional representation $\mathbf{Y}$.

Many of the weaknesses present in LLE are also present in Laplacian Eigenmaps (e.g. manifolds with discontinuities or holes). Despite this, many variants have been presented (e.g. He and Niyogi (2004); Sha and Saul (2005); Gerber et al. (2007); Jia et al. (2009)). Laplacian Eigenmaps also fits within the framework of spectral clustering (Ng et al., 2002) and graph partitioning (Mohar, 1991). As such, much of the theory behind these disciplines can be shared. This leads

[8]A simplified version exists where the weights are chosen such that $w_{ij} = 1$ if x_i and x_j are connected. This removes the need to select a kernel size t.

to Laplacian Eigenmaps being one of the most theoretically sound, and well understood, manifold learning algorithms.

Manifold Charting (2003)

Early research on manifold learning focused on two different approaches: local and global. Local methods attempted to preserve local properties of the manifold while global methods sought to preserve the global structure. Both approaches often failed at preserving their counterparts' properties (e.g. local methods failed at recovering the true global structure of the data). The next logical step therefore was to attempt to combine the strengths of both into a unified framework. This led to the development of the so called global alignment of local models algorithms.

Manifold Charting (Brand, 2003) was one of the first global alignment of local models techniques. The basic premise of Manifold Charting is to form a set of locally linear charts from the input data that are then stitched together into a global co-ordinate system in low-dimensional space. One of the unique aspects of the Manifold Charting approach to manifold learning is that it overcomes two of the big open problems in manifold learning. First, it provides the option to automatically estimate the intrinsic dimensionality of the manifold and it also allows new samples to be mapped back and forth from the high and low-dimensional spaces thus solving the out-of-sample extension problem[9].

Considering the estimation of the intrinsic dimensionality of the manifold as a separate process, then Manifold Charting has two main steps: 1) forming the local charts and 2) connecting these charts. The charting step seeks to recover the local information of the manifold with minimal loss whilst the connection step seeks to merge these charts into a globally consistent co-ordinate system.

[9]See Section 2.5.1 and Section 2.5.3 for discussions on these open problems.

The charts are formed by finding a soft partitioning of the data into locally linear low-dimensional neighbourhoods. Two criteria are set out to ensure that when connecting the charts a minimal amount of information is lost. These two criteria are (1) that there is a minimal loss of local variance and (2) there is maximal agreement of the projections of nearby points onto nearby neighbourhoods (Brand, 2003). These constraints are expressed as a posterior that combines a standard Gaussian Mixture Model (GMM) likelihood function with a set prior that penalises projections that are skewed with respect to existing co-ordinate frames. As shown in (Brand, 2003) criterion (1) is served by maximising the likelihood of a GMM density fitted to the data:

$$p(x_i|\mu, \Sigma) = \Sigma_j p(x_i|\mu_j, \Sigma_j)p_j = \Sigma_i \mathcal{N}(x_i; \mu_j, \Sigma_j)p_j \tag{2.31}$$

This means that each Gaussian component of the mixture model defines a local neighbourhood around μ_j with axes defined by the eigenvectors of the covariance matrix Σ_j. However this GMM is not sufficient to fully describe the data and as such a prior is needed. To understand this consider the example where two connected charts contain large subspace angles which leads to the inconsistent projection of a point between charts (that is the projection of a point given by one chart is widely different to the projection of the same point in the other chart). Criterion (2) seeks to minimise this by maximising the agreement of projections between charts. This is formulated in the prior:

$$p(\mu, \Sigma) = \exp[-\Sigma_{i \neq j} m_i(\mu_j) D(\mathcal{N}_i \parallel \mathcal{N}_j)] \tag{2.32}$$

where $m_i(\mu_j)$ is a measure of co-locality and $D(\mathcal{N}_i \parallel \mathcal{N}_j)$ is the cross-entropy between the two Gaussian models defined by the two neighbourhoods described as:

$$\begin{aligned}
D(\mathcal{N}_i \parallel \mathcal{N}_j) &= \int dx \mathcal{N}(x; \mu_i, \Sigma_i) \log \frac{\mathcal{N}(x; \mu_i \Sigma_i)}{\mathcal{N}(x; \mu_j \Sigma_j)} \\
&= (\log |\Sigma_i^{-1} \Sigma_j| \text{trace}(\Sigma_j^{-1} \Sigma_i) \\
&\quad + (\mu_j - \mu_i)^T \Sigma_j^{-1} (\mu_j - \mu_i) - D)/2
\end{aligned} \tag{2.33}$$

This cross-entropy can be thought of as describing the differences in size, orientation and position between the two co-ordinate frames described by the two mean values μ_i and μ_j and the axes specified by the eigenvectors of Σ_i and Σ_j.

Given the above soft partitioning of the data into a set of charts, manifold charting then attempts to stitch these charts together. Each chart is associated with an affine transform $\mathbf{G}_k \in \mathbb{R}^{(q+1) \times q}$ that projects the points of the k-th chart, $\mathbf{U}_k$, into a global co-ordinate space. For brevity we omit the full workings of the connecting step, a full description can be found in Section 4 of (Brand, 2003). The general idea of the connection step is to find the low-dimensional representation of a point by summing over all charts and computing the probability that a given chart generates the point. Generally, if two charts contain the same point then they should agree on the final low-dimensional position of that point.

One of the main disadvantages of Manifold Charting is its high computational complexity. Formulating manifold charting in terms of an optimisation problem, the cost function to solve is

$$\varphi(\mathbf{Y}) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{m} r_{ij} r_{ik} \parallel y_{ij} - y_{ik} \parallel^2 \tag{2.34}$$

where r_{ij} is the responsibility of point x_i to x_j as found by the Gaussian Mixture Models. This cost function can be re-expressed in terms of an eigen-problem but it, along with the GMM step, is still computationally expensive to execute.

Apart from its computational complexity Manifold Charting is a complete framework that solves many of the major problems in manifold learning such as the pre-image and out-of-sample extension problems. As such Manifold Charting lends itself well to many application areas and has been successfully applied to pose recognition (John et al., 2010).

Local Tangent Space Alignment (2004)

Local Tangent Space Alignment (LTSA) (Zhang and Zha, 2004) is another global alignment of local models technique in a similar vein to Manifold Charting. LTSA represents the local properties of the data in terms of local tangent spaces about each datapoint. These local tangent spaces are globally aligned to form a low-dimensional representation.

Core to the LTSA algorithm is the observation that, if local linearity is assumed, then there exists a linear mapping from a datapoint to its local tangent space in the high-dimensional space. Conversely, there exists a linear mapping from the corresponding low-dimensional datapoint to the same local tangent space (Zhang and Zha, 2004). As such, LTSA searches for the coordinates of the low-dimensional data at the same time as it searches for the linear mappings of the low-dimensional data to the tangent spaces of the high-dimensional data.

LTSA starts by finding the bases for the local tangent spaces at each datapoint. These bases are found by computing PCA on the k-nearest neighbours of each datapoint. This first step creates a local tangent space for each datapoint, Θ_i, and an associated linear mapping, $\mathbf{M}_i$, from x_i to Θ_i. As described above there also exists a mapping, $\mathbf{N}_i$, from the local tangent space co-ordinates to the low-dimensional co-ordinates. That is, $\mathbf{N}_i$ maps from $\theta_j \in \Theta_i$ to y_j. LTSA exploits this property and seeks to minimise the following objective function

$$\min_{y_i, \mathbf{L}_i} \sum_i \| \, y_i \mathbf{J}_k - \mathbf{L}_i \Theta_i \, \|^2 \tag{2.35}$$

where $\mathbf{J}_k$ is the centring matrix of size k. $\mathbf{L}_i$ is a linear mapping from the local tangent space to the low-dimensional space. The solution to this minimisation can be performed by eigendecomposition of an alignment matrix, $\mathbf{B}$, defined as the following iterative summation

$$\mathbf{B}_{\mathcal{N}_i \mathcal{N}_i} = \mathbf{B}_{\mathcal{N}_{i-1} \mathcal{N}_{i-1}} + \mathbf{J}_k (\mathbf{I} - \mathbf{V}_i \mathbf{V}_i^T) \mathbf{J}_k \tag{2.36}$$

for all matrices $\mathbf{V}_i$ starting from $b_{ij} = 0$ for $\forall_{ij}$. The low-dimensional representation can be obtained by computing the top q eigenvectors of the symmetric matrix $\frac{1}{2}(\mathbf{B} + \mathbf{B}^T)$.

LTSA has been extended to include a linear variant (Zhang et al., 2007) as well as an incremental version (Liu et al., 2006b). More recently an improved version of LTSA has been proposed that seeks to improve the tangent space construction in datasets with large curvature (Zhang et al., 2011). LTSA has been applied successfully to analysis of microarray data (Teng et al., 2005).

Global alignment of local models approaches ideally combine the strengths of local approaches with the strengths of global approaches. Conceptually, they also fit with the traditional view of a manifold which is modelled according to both its local metric information and the topology of these local neighbourhoods (Carmo, 1992). As such, it is likely that the best performing manifold learning algorithms will come from this family of techniques. It is for this reason that our developed method is a global alignment of local models algorithm.

Diffusion Maps (2006)

Diffusion Maps (DM) (Lafon and Lee, 2006) is a framework for dimensionality reduction, graph partitioning and data set parameterisation grounded in the field of dynamical systems. Central to the DM approach to manifold learning is the idea of diffusion distances. The diffusion distance is based on a Markov random walk on the neighbourhood graph of the data. Performing the random walk over a number of time steps the diffusion distance is obtained which defines the proximity of the datapoints. In theory, the diffusion distance is more robust to short circuiting than geodesic distances as it integrates over all paths through the graph (Maaten et al., 2009).

Once a neighbourhood graph of the data has been obtained, a weight matrix, $\mathbf{W}$ is constructed based on the approach presented by Laplacian Eigenmaps (Equation 2.28). A normalised form of this matrix, $\mathbf{P}^{(1)}$, is formed where the rows sum to 1. As such the entries of the matrix $\mathbf{P}^{(1)}$ are defined as

$$p_{ij}^{(1)} = \frac{w_{ij}}{\sum_k w_{ik}} \tag{2.37}$$

The Diffusion Maps framework considers this matrix to represent the probability of a transition from one datapoint to another in a single time step. The forward probability matrix for t time steps is therefore defined as $(\mathbf{P}^{(1)})^t$. Using this idea the diffusion distance can be defined as

$$D^{(t)}(x_i, x_j) = \sqrt{\sum_k \frac{(p_{ik}^{(t)} - p_{jk}^{(t)})^2}{\varphi(x_k)^0}} \tag{2.38}$$

where $\varphi(x_k)^0$ is a term that penalises parts of the graph with low density more than those with high density. Once the diffusion distances in the high-dimensional space have been found the low-dimensional representation tries to retain these diffusion distances. Lafon and Lee (2006) showed that the low-dimensional representation that retains the diffusion distances as closely as possible is formed by the eigenproblem

$$\mathbf{V}\mathbf{P}^{(t)} = \Lambda\mathbf{V} \tag{2.39}$$

where the top q columns of $\mathbf{V}$ ordered according to their associated eigenvalues in Λ represent the low-dimensional representation (the largest eigenvalue and eigenvector is discarded). The low-dimensional representation is then defined as

$$\mathbf{Y} = \{\lambda_2\mathbf{V}_2, \lambda_3\mathbf{V}_3, \ldots, \lambda_q\mathbf{V}_q\} \tag{2.40}$$

where the eigenvectors are normalised by their corresponding eigenvalues.

Diffusion Maps is an interesting approach to manifold learning as it contains within the framework more than just a manifold learning algorithm (e.g. graph partitioning). It is also based on a potentially more robust similarity measure than Isomap and therefore is less susceptible to short circuits in the graph. As such Diffusion Maps has been applied to research areas such as shape matching (Rajpoot et al., 2007) and gene expression analysis (Xu et al., 2007).

Other Algorithms & Artificial Neural Network Based Approaches

The above list of manifold learning algorithms is by no means exhaustive. Over the last ten years or so numerous new manifold learning algorithms, and extensions to existing manifold learning algorithms, have been presented. The above list touches on the algorithms that we believe are most important and most closely related to the work contained in this thesis. However, other interesting approaches to manifold learning do exist. Below we very briefly outline some of the more important of these approaches.

Maximum Variance Unfolding (MVU) (Weinberger et al., 2004; Weinberger and Saul, 2006b) can be seen as an extension to Kernel PCA that utilises Semidefinite Programming to 'unroll' the manifold. MVU is able to successfully learn a large class of image manifolds.

A class of algorithms also exist that model the underlying manifold as a Riemannian manifold (RML) (Brun et al., 2005; Lin et al., 2006). These algorithms achieve good results although it is unclear as to whether all manifolds can be correctly modelled as Riemannian manifolds.

Manifold Sculpting (MS) (Gashler et al., 2008) is an iterative approach using graduated optimisation that 'sculpts' the local neighbourhoods by simulating surface tension.

Verbeek presented a method for manifold learning that uses a global alignment of local factor analysers (CFA) (Verbeek, 2006). As such it has much in common with Manifold Charting (Brand, 2003) and Local Tangent Space Alignment (Zhang and Zha, 2004), however it has more in common with self-organising approaches such as SOM (Kohonen, 1995) and GTM (Bishop et al., 1998).

Another global alignment technique is Local Linear Coordination (LLC) (Roweis et al., 2002) which uses a probabilistic framework to find the global alignment of a mixture of factor analysers.

Greedy Procrustes (GP) (Goldberg and Ritov, 2009) seeks to find a low dimensional embedding by calculating the Procrustes transformations of increasing local neighbourhoods. Starting at a random datapoint the local neighbourhood is embedded into low-dimensional space using PCA. The embedding is then found by performing PCA embedding followed by Procrustes alignment on all neighbourhoods of previously embedded points.

Finally, a recent linear technique based on approximately harmonic projections, AHP, presents a novel direction in the field of manifold learning (Lin et al., 2010). It is based on the observation that the manifold to be learnt is often split into numbers of separate connected components. As such, AHP attempts to separate these distinct connected components using a harmonic framework.

2.3.3 Artificial Neural Network Based Approaches

The above discussions on manifold learning purposefully omit a large class of manifold learning and dimensionality reduction algorithms: those based on artificial neural networks. This class of algorithm includes Self-Organising Maps (Kohonen, 1995), Generative Topographic Mappings (GTM) (Bishop et al., 1998) and Multilayer Autoencoders (Hinton and Salakhutdinov, 2006). The main reason behind omitting these approaches is that they present such a very different paradigm to manifold learning than those previously discussed. These algorithms utilise neural networks to obtain the low-dimensional embeddings and so very often represent 'black boxes' into which high-dimensional data is passed and low-dimensional data are produced. As such, these methods violate one of the core criteria that we set out in Chapter 1 for our manifold learning algorithm to meet (ease of understanding). Also, methods such as SOM and GTM can be thought of more as clustering algorithms than manifold learning algorithms. A discussion of the link between dimensionality reduction and clustering algorithms can be found in (Bengio et al., 2004).

	Manifold	Feature	Scale
PCA	Linear	Data Covariance	Global
Isomap	Non-Linear	Geodesic Distances	Global
LLE	Non-Linear	Local Fits	Local
LTSA	Non-Linear	Local Tangent Planes	Local/Global
AHP	Linear	Harmonic Projections	Global
MDS	Linear	Euclidean Distance	Global
Sammon Maps	Linear	Euclidean Distance	Global
DM	Non-Linear	Diffusion Distance	Global
Kernel PCA	Non-Linear	Kernel Product	Global
MVU	Non-Linear	Kernel Product	Global
RML	Non-Linear	Riemannian Co-ordinates	Global
Eigenmaps	Non-Linear	Weighted Graph Distance	Local
GP	Non-Linear	Procrustes Metric	Local
MS	Non-Linear	Local Tension	Local
SNE	Non-Linear	Local Probabilities	Local
Charting	Non-Linear	GMM Charts	Local/Global
CFA	Non-Linear	Local Factor Analysers	Local/Global
LLC	Non-Linear	Local Factor Analysers	Local/Global

Table 2.1: A taxonomy of manifold learning techniques. The first four techniques are those included in the analysis chapter of this thesis. The techniques in the latter half of the table are those considered in Section 2.3.

2.4 Towards a Taxonomy of Techniques

Providing a high-level taxonomy of manifold learning algorithms is a useful way of differentiating and distinguishing between the different approaches (Lee and Verleysen, 2007; Maaten et al., 2009). Table 2.1 shows one such taxonomy where the algorithms are described according to the class of manifold they attempt to learn (linear or non-linear), the features they attempt to retain between the high and low-dimensional spaces (e.g. Euclidean distances, Geodesic distances), and also the scale at which these features are considered (local, global or local & global).

Our developed approach fits into the above taxonomy as a non-linear, local and global technique that attempts to preserve local PCA models across a global scale.

2.5 Open Problems in Manifold Learning

Although manifold learning is becoming an established research field there are still many unsolved problems associated with the area. These 'open problems' are interesting areas which arguably deserve as much attention as the manifold learning algorithms themselves. For manifold learning algorithms to become truly useful many of these open problems will need to be tackled. However, at the time of writing many of them are still unresolved. We outline some of the major open problems in the field of manifold learning and discuss some of the current state of the art attempts at solving them.

2.5.1 Intrinsic Dimensionality

Intrinsic dimensionality is concerned with the dimensionality of the manifold rather than the dimensionality of the space within which it resides. Intrinsic dimensionality estimators can be used to gain an estimate as to the dimensionality of a manifold contained within a high-dimensional space. This process is important when the target dimensionality of a dataset is unknown as we can estimate the optimal target dimensionality to embed the data in.

Simplistic approaches to intrinsic dimensionality are based on the local properties of the data. Two such approaches were presented in (Verveer and Duin, 1995) and are based on the analysis of small, local, regions of the manifold.

The local eigenvalue algorithm presented in (Verveer and Duin, 1995) was first discussed in (Fukunaga and Olsen, 1971) and is based on observing the largest change between eigenvalues of small regions on the manifold. Imagining a graph of the eigenvalues the intrinsic dimension based on this method is equal to the largest 'jump' between points on the graph. The second method described in (Verveer and Duin, 1995) is an extension of the nearest neighbour algorithm first presented in (Pettis et al., 1979) which examines the distribution of data points over local neighbourhoods to estimate the local intrinsic dimensionality.

A popular global approach to estimate the intrinsic dimensionality of a dataset is to investigate the correlation dimension of the data (Grassberger and Procaccia, 1983; Camastra and Vinciarelli, 2002). The method uses concepts from fractal and chaos theory to obtain an estimate as to the intrinsic dimensionality of the data.

Since the advent of manifold learning algorithms and non-linear dimensionality reduction techniques, more advanced intrinsic dimensionality estimators have been proposed. These more recent methods attempt to take the non-linearity of the manifold into account. Kégl presented a method based on the packing dimension of the data (Kégl, 2002). This method does not rely on any parameters and is able to exploit the underlying geometric structure of the data to obtain an estimate as to its intrinsic dimensionality. Levina and Bickel suggested an approach that estimates the intrinsic dimensionality of the manifold based on the Maximum Likelihood Estimation of a Poisson process approximation (Levina and Bickel, 2004). A graph based technique was presented in (Costa and Hero, 2004) that bases its intrinsic dimensionality estimation on the growth rate of the geodesic total edge length functional of entropic graphs. Finally, an interesting geometric approach was discussed in (Fan et al., 2009) where the estimation of intrinsic dimension is obtained by examining the relationship between local incising balls and the number of samples contained within these balls.

As with manifold learning, there is no gold standard intrinsic dimensionality estimator. Often the simple local linear approaches provide good approximations as to the intrinsic dimensionality of the data and more often than not will suffice for most applications. The newer, more advanced, estimators are often too complex to be used on large datasets. A possible approach to obtaining a good estimate as to the intrinsic dimensionality of the data is to take an average of various intrinsic dimensionality estimators and either take the mean value, or to obtain a more robust estimate, and given enough estimators, take the median value.

One other point to note, and a possible further research direction, is that none of the above methods account for manifolds with variable intrinsic dimensionality. In some applications and for some data, it seems unreasonable to model the entire manifold with the same intrinsic dimension. As such it would be interesting to investigate the idea of estimating a manifold with multiple intrinsic dimensions.

2.5.2 Parameter Estimation

Many manifold learning algorithms use parameters to help adapt their solutions to different datasets. For example, the algorithm setup for a very high-dimensional dense dataset will be different to that of a sparsely sampled low-dimensional dataset. Most manifold learning algorithms take a single parameter, neighbourhood size[10]. The majority of work on the correct parameter value selection for manifold learning algorithms is focused on this neighbourhood size parameter.

[10]The target dimensionality is another parameter, but here we assume that this is known.

The neighbourhood size parameter, k, is used in two different settings in manifold learning: either to construct the neighbourhood graph between all data points or to determine the size of local models. In each of these cases incorrect neighbourhood sizes can lead to detrimental results. When building a neighbourhood graph, if the value of k is too large then short-circuits will occur in the graph causing parts of the manifold that are distant in the high-dimensional space to be embedded closely in the low-dimensional space. Similarly, if the value of k is too small then a disconnected graph could be formed meaning that the manifold is incorrectly modelled as a set of disjoint sub-manifolds. When k is used to build local models then the effects of incorrect k values are different. If the value of k is too small then the manifold will be under sampled, meaning that not enough local information is available to reconstruct the local properties of the data. If the value of k is too large then the manifold will be over sampled leading to a possible break in the local linearity assumption (that is, the local model covers a large patch on the manifold that contains non-linear structure).

When considering the spectral case (i.e. building a k-neighbourhood graph) techniques exist to help estimate the optimal parameter size. Two similar methods were presented to find the optimal parameter value when using LLE (Kouropteva et al., 2002) and Isomap (Samko et al., 2006). More general techniques to find the k value specifically for spectral graph techniques have also been presented (Yang, 2005; Geng et al., 2005; Wen et al., 2007; Lewandowski et al., 2010).

Fewer methods exist to estimate the correct neighbourhood size for model formation. Wang et al. (2004) presented an algorithm based on the local tangent space around each datapoint that selects the optimal parameter by examining the ratios of the square of singular values. However, it does require a global threshold value, so it could be argued that it does not fully solve the problem.

Another threshold based approach was presented in (Nathan and John, 2006) where the sampling density of the local tangent space is used to obtain a parameter estimate. If the projection of the near points in the direction of the normal onto the local tangent space is less than the threshold, then they are selected as neighbourhood points. This method does however rely on the assumption that the sampling conditions are ideal therefore making it unusable in many real settings (Gao and Liang, 2011). A similar approach based on the angle to the tangent subspace is presented in (Karina et al., 2007).

A more recent algorithm has been presented by Gao and Liang that attempts to bridge the gap between both spectral and model building estimation techniques (Gao and Liang, 2011). Their method uses dynamical neighbourhoods around each point to build the local tangent spaces as well as neighbourhood graphs. As such it addresses the problem of non-uniform neighbourhood sizes, which is a problem many of the above algorithms do not attempt to tackle. The strength of the non-uniform approach is that different regions of the manifold can have different neighbourhood size parameters making manifold learning algorithms more robust to changes in sampling of the data.

Although techniques exist for estimating the optimal parameters for manifold learning algorithms it is still worthwhile to investigate the performance of an algorithm over a range of parameter values. This will then show how the algorithm responds to different parameter values as well as indicating an optimal range of values within which the algorithm can work for the given data set.

2.5.3 Out-of-Sample Extension

The out-of-sample extension problem is concerned with the projection of novel data samples into previously learnt embeddings. If we have previously run a manifold learning algorithm on a set of data samples and are given a novel 'unseen' sample, how do we correctly embed this sample into the low-dimensional space without re-learning the entire manifold?

In the simplest case, if the manifold has been learnt by a linear technique such as PCA, then we can obtain an estimate of a new point's low-dimensional position by projecting it onto the low-dimensional basis vectors. However, as previously shown most manifold learning algorithms are not based on a linear transformation and as such there is no simple solution to the out-of-sample extension problem.

Some manifold learning techniques come already equipped with the ability to handle novel data samples. All linear techniques, such as PCA (Hotelling, 1933), MDS (Torgerson, 1952; Cox and Cox, 2001) and LPP (He and Niyogi, 2004), inherently allow for novel samples to be added to the low-dimensional space by projecting onto the linear basis learnt by the manifold learning algorithm. Some non-linear techniques also allow for new samples to be added. For example, Manifold Charting (Brand, 2003), Co-ordinated Factor Analysis (Verbeek, 2006) and Kernel PCA (Scholkopf et al., 1998).

Very little research has gone into the generalisation of the out-of-sample extension problem. Bengio et al. (2003) extended Isomap, LLE, Laplacian Eigenmaps and MDS to allow for the out-of-sample problem by phrasing them within a common kernel framework. Yang et al. then further extended this idea by presenting a generalised kernel framework exploiting a regularisation term to allow for a generalised out-of-sample extension (Yang et al., 2010). In Chapter 5 we discuss the out-of-sample extension in more detail and present a generalised solution to the out-of-sample extension problem based on the local geometry change of the manifold between the high and low-dimensional spaces.

A related problem to the out-of-sample extension is that of the pre-image problem. This can conceptually be thought of as the reversal of the out-of-sample extension. Given a novel point in low-dimensional space, the pre-image problem is concerned with learning its position in the high-dimensional space. The pre-image problem is a well studied problem in kernel methods (Kwok and Tsang, 2004; Bakir et al., 2004). Often the pre-image problem can be used to help remove noise from images and shapes (Thorstensen et al., 2009).

Although methods have been developed to help solve the out-of-sample and pre-image problems, there is no single framework that can solve both problems in a continuous manner (i.e. points can be mapped to and from the high and low-dimensional spaces). The creation of such a framework could prove useful for areas such as manifold based classification (Jun and Ghosh, 2010).

2.5.4 Large Scale Data

One of the drawbacks of most non-linear dimensionality reduction and manifold learning techniques is their large computational complexity and space requirements. This means that in their native form they are unsuitable for learning large data sets. When we discuss large datasets we mean large in two ways: large numbers of samples and large numbers of dimensions. Typically large datasets will have $n \geq 2000$ and $p > 50$ (Lee and Verleysen, 2007). One approach to help overcome this problem is to subsample the data. However, this could introduce further problems rather than solve the large scale learning problem (See Section 2.5.5). A more rigorous approach was presented in (Talwalkar et al., 2008). They tackle the problem of spectral decomposition of a large matrix. Spectral decomposition is the heart of many manifold learning algorithms and so a solution to this problem could help overcome one of the main bottlenecks in manifold learning. Their approach investigates two approaches to large scale

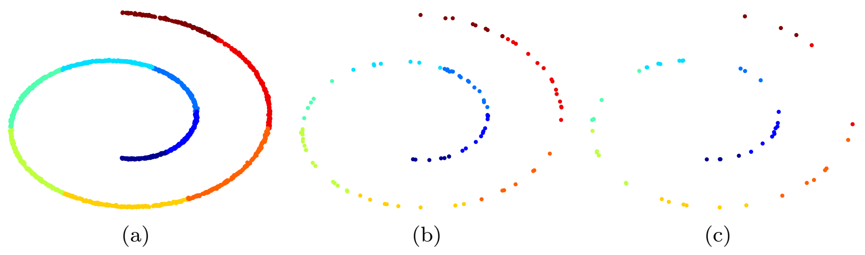

(a) (b) (c)

Figure 2.3: The effect of sparsity on the ability to recover the true low-dimensional structure of a dataset. As the sparsity increases the manifold becomes harder to identify and discontinuities in the sampling will lead to disconnected components in the neighbourhood graph or incorrect inter-point distances being calculated.

decompositions: Nyström approximation and column sampling. They test their method on a large scale (18 million data points) image database and achieve impressive results.

Although the method described in (Talwalkar et al., 2008) is a step towards providing a solution to the large scale problem it is still someway off being truly usable[11]. As such, the large scale problem remains one of the most under-researched, yet important open problems. If manifold learning is to be used in real world, large scale applications, such as bio-medical image analysis, remote sensing, and data mining, then methods need to exist to scale manifold learning algorithms to effectively handle large data.

2.5.5 Sparsity

As discussed in Section 2.1, as the dimensionality of the space increases so does the sparsity of the data. Often as the dimensionality of the space increases more data points are needed to properly reconstruct the manifold. However, this is usually not possible and as such sparsity is a problem which manifold learning algorithms will need to be able to handle effectively.

[11]Their results were obtained using a cluster of several hundred machines.

As an example of this, Figure 2.3 shows the 2-dimensional Swiss Roll dataset with different numbers of samples drawn from the original parameterisation. In the densely sampled case (Figure 2.3 (a)) the curved structure of the manifold is visually apparent and as such it is expected that most manifold learning algorithms would be able to successfully learn the low-dimensional embedding. However, as the density decreases and sparsity increases (Figure 2.3 (b-c)) the structure becomes less well defined. As such discontinuities appear in the manifold and the low-dimensional embedding will become harder to learn.

The question of how to deal with sparse data from a manifold learning perspective remains an open one. One interesting approach to dealing with sparsity is found in (Silva et al., 2005) where the sparsity of the data is exploited to improve the performance of some manifold learning algorithms. Using LASSO regression the manifold is built from sparse landmark points. This could be an interesting starting point for further research in this area.

Although many manifold learning algorithms will be unable to learn meaningful embeddings for very sparse data, it is still desirable for such embeddings to retain some structure and for them not to completely collapse in on themselves. As such, it is expected that the embeddings produced using good manifold learning algorithms will degrade gracefully as the sparsity of the data increases. This is in contrast to the performance of poor manifold learning algorithms where a sudden drop in performance would be observed at a certain sparsity level.

2.5.6 Quality Assessment

In the above discussions mention was made as to a manifold learning algorithm producing a ‘good’ or ‘faithful’ embedding. The question is then raised as to what constitutes a ‘good’ embedding? This area of quality assessment is a difficult one as, more often than not, no ground truth embeddings are available. It is therefore impossible to measure how far the embedding produced by a manifold learning algorithm deviates from the 'perfect' embedding. As such, the ground truth needs to be somehow estimated at either a global or local scale, or at both local and global scales.

Residual variance is often used to measure the global stability of an embedding (Tenenbaum et al., 2000). The residual variance is measured by calculating the correlation co-efficient between the matrix of distances in the high-dimensional space and the matrix of Euclidean distances in the low-dimensional embedding. The high-dimensional distance matrix is usually calculated by measuring the geodesic distances between points along a neighbourhood graph[12]. The geodesics are often difficult to calculate and, as described in Section 2.5.2, the selection of the correct neighbourhood size parameter is still an open problem.

The local stability of an embedding is often easier to assess as the manifold in high-dimensional space can be thought of as locally linear. Such methods for measuring the local stability include Trustworthiness & Continuity (Venna and Kaski, 2006) and the Procrustes Error (Goldberg and Ritov, 2009). These methods work by measuring the change in local neighbourhoods between the high and low-dimensional spaces. Trustworthiness and Continuity measure the change in neighbourhood relations (i.e. points entering and leaving a neighbourhood as a result of manifold learning) and can be thought of as a topological measure as they ignore metric information. The Procrustes Error on the other hand can be thought of as a geometric measure as it calculates the rotational and scale change between neighbourhoods in the high and low-dimensional space.

An attempt to extend the rank based criteria methods (such as Trustworthiness

[12]As such the Isomap algorithm (Tenenbaum et al., 2000) can be thought to minimise this correlation co-efficient.

and Continuity) into a more robust framework was made in (Lee and Verleysen, 2009). They proposed the creation of a co-ranking matrix to compare the ranks of data points between the high and low-dimensional spaces. However, the results produced are often difficult to interpret.

Although various quality assessment criteria exist, there is no 'one size fits all' measure, and as such when assessing the quality of an embedding it is often worth measuring a variety of different criteria. A good strategy would be to use a global approach such as Residual Variance, to measure the global stability, and then local approaches to measure the topological and geometric stability.

2.6 Further Reading

This chapter provides a brief overview of the area of manifold learning as well as the foundations and problems associated with it. Therefore, we now point the interested reader to further sources of information related to the field of manifold learning and non-linear dimensionality reduction. A good introductory book to the field of manifold learning is "Nonlinear Dimensionality Reduction" by Lee and Verleysen (Lee and Verleysen, 2007). It provides a detailed introduction to the field of manifold learning and discusses some of the main algorithms in the field. What is of potentially more interest is the final chapters and appendices where an outline for the correct usage of manifold learning is given. Review papers are also available that contain overviews of the field of manifold learning and dimensionality reduction (e.g. Maaten et al. (2009); Fodor (2002); Burges (2004)).

An excellent online lecture on Geometric Methods and Manifold Learning delivered by the Mikhail Belkin and the late Partha Niyogi at the 2009 Machine Learning Summer School is available at `http://bit.ly/oShCNZ` (Link checked

25/07/2011).

2.7 Conclusions

From this work we have found that global alignment of local models techniques are, to us, the most interesting manifold learning algorithms as they attempt to capture both the local and global properties of the data. As discussed in Section 2.3.2.5 they are also theoretically grounded. The traditional view of a manifold is that it can be modelled according to both its local metric information and the topology of these local neighbourhoods (Carmo, 1992). As such, our developed method, Piecewise-Linear Manifold Learning, is a global alignment of local models technique. We attempt to capture and maintain the local information of the manifold via local PCA models, whilst also providing a globally aligned representation of these models found according to their topology.

We have also decided to provide a generalised solution to the out-of-sample extension problem. As shown in Section 2.5, this is one of the open problems in manifold learning and little work has been carried out into attempting to provide a robust generalised solution to this problem. The solution to the out-of-sample extension problem allows novel data points to be quickly embedded into previously learnt embeddings without the entire manifold needing to be relearnt. We believe that a generalised solution to this problem will help manifold learning algorithms to be used in more real world tasks such as classification.

"Young man, in mathematics you don't understand things. You just get used to them."

John von Neumann (1903-1957)

3 Piecewise-Linear Manifold Learning[1]

Piecewise-Linear Manifold Learning (PLML) is a new approach to manifold learning that builds on the strengths of linear techniques such as Principal Components Analysis (Hotelling, 1933) and also recent advances in global alignment of local linear models (e.g. Local Tangent Space Analysis (Zhang and Zha, 2004) and Manifold Charting (Brand, 2003)). The core of the PLML algorithm is the merging of local models according to their topology as defined by a Minimum Spanning Tree. Unlike existing approaches where a global alignment is attained by statistical methods (e.g. exploiting the local tangent space (Zhang and Zha, 2004)) the PLML approach is piecewise such that the process can be paused and

[1]

Harry Strange and Reyer Zwiggelaar. Iterative Hyperplane Merging: A Framework for Manifold Learning. In Proceedings of the British Machine Vision Conference, pages 18.1-18.11. BMVA Press, September 2010

Reyer Zwiggelaar and Harry Strange(2010) Patent KS.P47544GBi

Harry Strange and Reyer Zwiggelaar. Parallel Projections for Manifold Learning. In Proceedings of the Ninth International Conference on Machine Learning and Applications. Washington DC, December 2010. IEEE Press.

examined at any step and proceeds in a defined manner. This makes it easier to understand the embedding produced by the PLML algorithm and also makes it suitable for larger scale applications. This chapter is only concerned with an explanation of how the PLML algorithm works, for a discussion of experimental results see Chapter 5.

In the rest of this chapter we introduce the Piecewise-Linear Manifold Learning algorithm at both a high and low level. Section 3.1 provides a high-level description of the algorithm and is suggested reading if the reader only wishes to gain a basic understanding of how the algorithm operates. We end this section by performing a simple step through of the algorithm in Section 3.1.5. In Section 3.2 we describe the PLML algorithm in more depth and provide the mathematical basis for the algorithm. Finally, we end in Section 3.3 by outlining the alternative research paths that were investigated during the development of the PLML algorithm.

3.1 PLML Framework Overview

In this section we aim not only to provide a high-level overview of the Piecewise-Linear Manifold Learning (PLML) algorithm but also to outline the key similarities and differences our approach has to existing algorithms.

The PLML algorithm can be coarsely split into 4 main parts:

1. Building local linear models of the input manifold.

2. Determining the topology between the local models.

3. Combining the local models according to the topology to form a globally consistent alignment.

4. Obtaining a low-dimensional embedding of the global alignment of local models.

Each of these steps is described in more detail later in this chapter but for now we will provide a brief description of each as we trace our way through the algorithm.

3.1.1 Step 1: Building Local Linear Models

The first step in the algorithm is to form the local models from the input data. These local models will then be the blocks upon which the rest of the algorithm builds. The aim of this first step is to create a set of models from the input data that contain a set number of data points and are locally linear. This is shown by the example given in Figure 3.1. The data is partitioned into discrete models using a constrained clustering algorithm (explained in more detail in Section 3.2.1). Once the data has been partitioned these local clusters are turned into local hyperplanes by projecting them onto their basis vectors (found by performing PCA on the data points within each cluster). These local hyperplanes are now locally linear but still maintain their position on the manifold in high-dimensional space. Throughout the rest of this section we

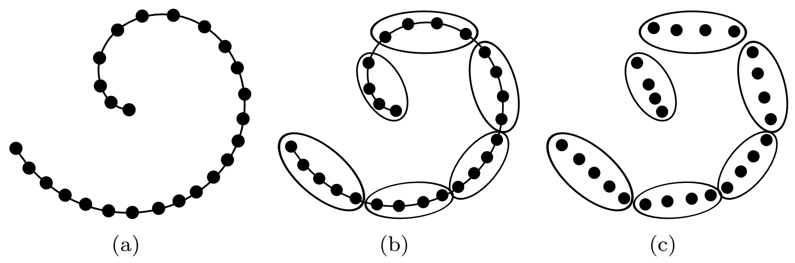


Figure 3.1: A simple data set sampled from a curved surface (a). The first step of the PLML algorithm is to cluster the data (b) and then project the points within each cluster onto their principal components forming local hyperplanes (c).

refer to these local hyperplanes as local models. At a local model scale the data is low-dimensional, that is if we were to examine each model individually the data would lie on a low-dimensional hyperplane whose dimensionality at a local scale is equal to that of the low-dimensional space into which we are wishing to embed our data. Using the example in Figure 3.1 (c) the data within each model lies on a straight line. However, each model has its own coordinate system. The local models may be locally low-dimensional but they do not lie in a global low-dimensional co-ordinate space. For example if you were to take two models from Figure 3.1 and place them side by side they would not both lie on the same straight line so they do not both lie in the same low-dimensional co-ordinate space. This problem is known as the global alignment problem: how can we form a global co-ordinate system from the local low-dimensional models? The remainder of the PLML algorithm is concerned with this global alignment problem.

This problem of globally aligning local models is one much studied in the literature. Local PCA (Kambhatla and Leen, 1994) can be thought of as the first truly local dimensionality reduction technique but it did not attempt to join the local PCA models into a global coordinate system. LLE (Roweis and Saul, 2000) creates local models of the data based on their linear re-constructions but the global shape of the manifold is not successfully reconstructed as there is often not enough global information contained in the local reconstructions to obtain a good estimate of the global manifold. Global alignment techniques such as LTSA (Zhang and Zha, 2004) and Manifold Charting (Brand, 2003) improve upon this by approximating some form of global alignment of the local models using either the optimisation of the local tangent spaces (Zhang and Zha, 2004) or using charting of Gaussian Mixture Models (Brand, 2003). Our approach differs from existing global alignment methods as it seeks to learn the

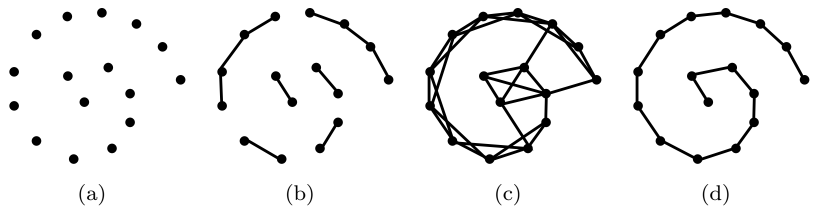


Figure 3.2: An example of the differences between a k-neighbour graph that is too small (b), too large (c) and the Minimum Spanning Tree (d).

global topology between the local models and use this topology as the basis for combining the local models into a global coordinate system.

3.1.2 Step 2: Determining Model Topology

The question of how best to determine the topology between the local models is an interesting and difficult one and is analogous to finding the correct topology of a data set[2]. The goal is to determine how the local models formed in Step 1 are connected (i.e. which models are neighbours and which are not). A simple method would be to employ a k-nearest neighbour approach, that is to define the k-nearest models as neighbours. However as discussed in Chapter 2 the k-nearest neighbour graph of a data set can often ill approximate the connectivity by either over connecting the data (leading to short circuits in the topology) or by under connecting the data (leading to disconnected sub graphs). A more robust method than the k-nearest neighbour graph is to use the Minimum Spanning Tree (MST) of the data to describe its topology (Figure 3.2). Robins showed (Robins, 2000) that the MST of a dataset provides a good approximation of its underlying topology. The MST contains all the information needed to describe the global connectivity of a dataset and since topology can be, at least simplistically, thought of as the connectivity of data then the MST

[2]Various approaches to determining data topology have been presented, e.g. (Aupetite, 2006), (Silva and Carlsson, 2004), (Robins et al., 2000).

is a good way of approximating the underlying topology. The MST of the local models therefore provides us with an estimate of the connectivity. Through the use of the MST we have information as to which models are connected in a nearest neighbour sense and which are not.

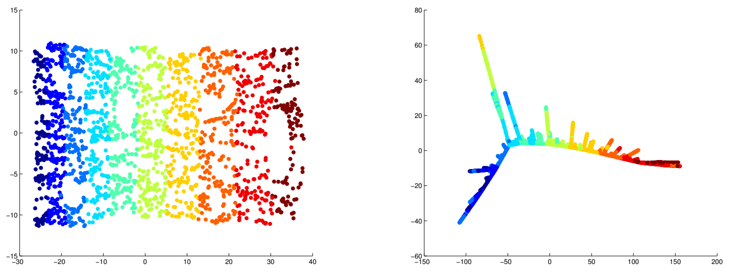
However, not all data sets will exhibit a tree like topology as captured and represented by the Minimum Spanning Tree. The example shown in Figure 3.2 has a tree like topology, however a data set could easily be imagined that does not contain such a structure. For example, the data could contain local sections which, at a point level, are considered neighbours but due to the tree structure of the MST are not neighbours in the MST. A neighbourhood graph could potentially overcome this limitation, but some of the important properties of the MST as described above would not necessarily be present. As such, the alignment step described later on in this algorithm needs to be able to handle this potential lack of connectivity. If the alignment step is done properly, and distortions and mis-alignments are kept to a minimum, then the fact that two nearby models are not connected by the MST will not matter as the alignment algorithm will correctly place them anyway.

Since the MST provides a good approximation of the data topology then one might assume that it provides a good alternative to the k-neighbourhood graph in spectral techniques such as Isomap (Tenenbaum et al., 2000). An attempt was made by Roychowdhury (Roychowdhury and Ghosh, 2009) to use the MST as a replacement for the k-neighbourhood graph in Laplacian Eigenmaps (Belkin and Niyogi, 2002) to better reconstruct global information. Although the MST does contain topological information and it can be used to roughly reconstruct the global shape of the manifold it does not contain enough proximity information to yield useful results (See Figure 3.3). There is not enough local information contained within the MST to successfully reconstruct the manifold. A more robust approach was presented by Carreira-Perpinán in (Carreira-Perpinán, 2005) where the MST is calculated over many iterations with jitter (e.g. noise) added at each iteration to produce slightly different graphs each time. The sum of these graphs then provides a good approximation of the neighbourhood graph. However, in the PLML algorithm the MST is not used as a normal neighbourhood graph as in most spectral techniques.

As well as the topological properties of the MST other properties exist that make it suitable for our application, the main one being its non-cyclic nature which makes it useful for traversal. We exploit the topological stability of the MST to describe how the models are connected and we use the non-cyclic traversal property to connect the local models.

3.1.3 Step 3: Merging Models

At a simple level the models are connected by merging them together in an ordered fashion to create a global 'merged' model. This merging process is one of the key steps of the PLML algorithm but before we discuss this process in



(a) ISO-k (b) ISO-MST

Figure 3.3: An example of the differences between using a k-neighbour graph for Isomap (a) and using an MST (b) . Although the MST manages to uncover the principal directions of the manifold (shown by the transition of labelling from blue to red) it is unable to map the manifold structure correctly.

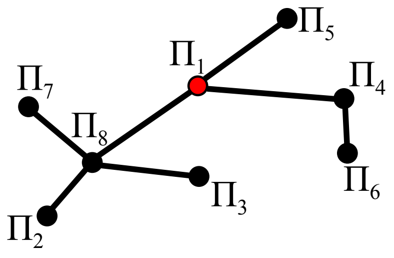


Figure 3.4: An in-order traversal of the MST of the local models (Π_1-Π_8) allows us to find the correct order that the models should be visited. See body text for full description.

detail we show how the MST can be used to define the ordering with which the models are to be merged. The MST provides us with the topology and therefore the order with which we can connect the local models and we can exploit this ordering via a pre-order traversal. A pre-order traversal from any node in the MST will provide us with the order in which the models should be visited. Consider the simple example shown in Figure 3.4. In this example there are 8 models labelled Π_1 to Π_8. We begin at model Π_1 and perform a pre-order traversal to all other models. Given this we know that the traversal order will be:

$$\text{order} = \{\Pi_1, \Pi_8, \Pi_7, \Pi_8, \Pi_2, \Pi_8, \Pi_3, \Pi_8, \Pi_1, \Pi_5, \Pi_1, \Pi_4, \Pi_6\} \tag{3.1}$$

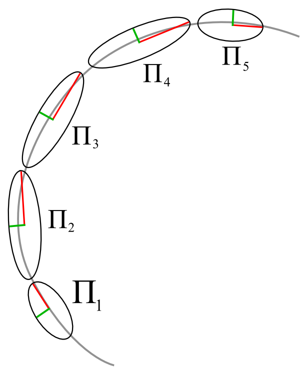
It is worth noting that this ordering includes movement forward (to previously unvisited models) and backward (revisiting models). Since we wish to find a global model by merging the local models this ordering only describes half the process. When moving from one model to a new unvisited model according to the above ordering we form a completely new model that is based on the combination of the two models. So, given the above example when we reach model Π_3 we have already visited models Π_1, Π_8, Π_8 and Π_2. Therefore we have a global model Π_{global} that is the merged product of all the previously visited models, that is

$$\Pi_{\text{global}} = \Pi'_8 \oplus \Pi'_2 \oplus \Pi'_7 \oplus \Pi'_1 \tag{3.2}$$

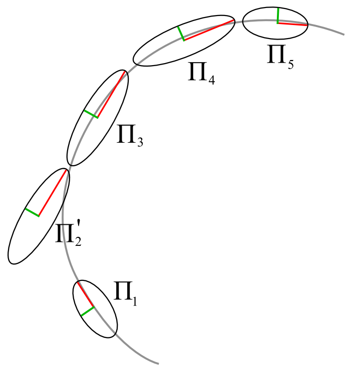
where $\oplus$ refers to the merging process and Π'_i refers to the merged version of the model Π_i.

So the merging process is not simply the merging of the current model with the previous model but rather the merging of the current model with the *merged representation of all the previously visited models*. This gives rise to the concept of the global model. The global model can be thought of as the merged representation of the previously visited models (this is because when two models are merged a new merged representation of the models is created rather than modifying the existing model representations).

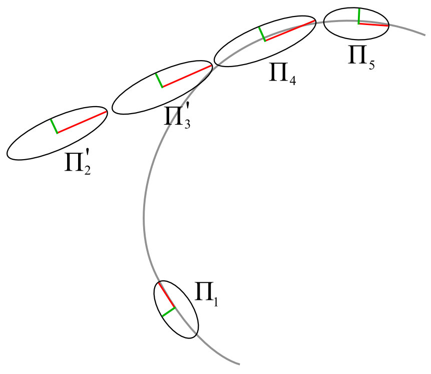
The task now becomes how to merge the models during this traversal so as to obtain a global approximation of the manifold. This merging process is done by building an increasingly global model during the traversal of the models as described above. When moving from one model to the next we merge the models by projecting the previously visited models onto the hyperplane of the current model. This process is briefly outlined in Figure 3.5. Starting the traversal of the MST at any model we then move to the next model in the traversal order and merge the two models. This merging equates to projecting the previously visited model onto the hyperplane defined by the current model. In the example in Figure 3.5 the model Π_2 is projected onto the hyperplane defined by Π_3 and the global model consists of Π'_2 (the merged representation of Π_2) and Π_3. Continuing this, when we move on to the next model in the traversal order, Π_4,



(a)



(b)

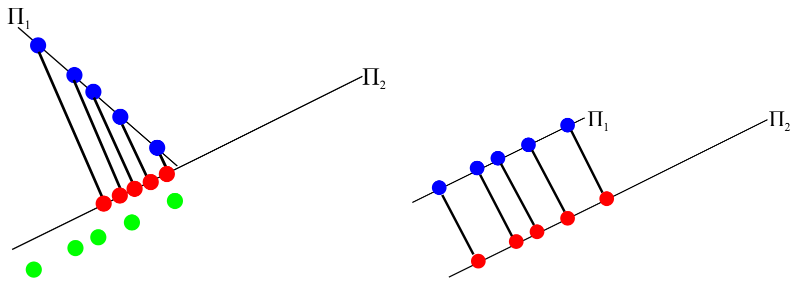


(c)

Figure 3.5: An example of how the merging process works in the PLML algorithm. We begin at model Π_2 and then merge with model Π_3. When moving on to model Π_4 the combined merged representations of Π_2 and Π_3 (denoted Π_2' and Π_3') are merged onto Π_4.

we project the global model onto the hyperplane of Π_4. At this stage the global model consists of Π'_2, Π'_3 and Π_4. This merging process is continued until all models have been reached according to the MST traversal order.

At the heart of this merging process is the projection of the global model onto the current hyperplane. This hyperplane projection is defined by a subspace projection followed by a translation. The subspace projection ensures that the global model lies on the basis vectors of the current hyperplane (that is they exist in the same co-ordinate space) and the translation moves the global model onto the current model's hyperplane. The translation step is trivial, we simply project a known point on the current model onto the basis vectors and define the translation as the difference between the two (discussed in more detail in Section 3.2). Of more interest is the projection step. Subspace projections are always orthogonal to the basis vectors of the subspace being projected onto. This becomes problematic if the angle between the two models is large (Figure 3.6 (a)). If the dihedral angle between the two hyperplanes defined by the two models is large then distortions will be introduced as a result of the projection. Due to the iterative nature of the PLML algorithm this distortion will continu-



(a) orthogonal projection (b) parallel projection

Figure 3.6: (a) An example of an orthogonal projection between two hyperplanes with a large dihedral angle leading to distortions, shown by the difference between the red projected points and the true representation of these points shown in green. (b)The same hyperplanes but with Π_1 rotated so that the projection is parallel and the dihedral angle is 0.

ally degrade the global model leading to an unusable result[3]. To improve on this we rotate the global model so that it is parallel to the current model meaning that the projection is parallel (Figure 3.6 (b)). This reduces the distortion as a result of projection and adds little to the computational cost of the algorithm (a more detailed discussion of how this rotation is performed can be found in Section 3.2).

One other special case that needs to be considered is when moving back along the MST during traversal. Just as the global model moves forward with each forward step in the traversal it also needs to move back with each backward step. Rather than repeating the projection problem described above we only need to perform a simple alignment of models. This is due to the fact that when visiting a model during the backtracking process we already have an image of this model in the global alignment. So the back tracking step becomes aligning the model's representation in the global model to the current model. This alignment can be found by running the Procrustes algorithm (Gower, 1975) to find the rotation, translation and scale transformation to match the model's representation in the global hyperplane to its original representation.

3.1.4 Step 4: Finding the Low-Dimensional Embedding

Once the traversal has completed we know that we have a global alignment of the local models. The final step of the process is to reduce the dimensionality of the global model. Since we know that it is linear to the low-dimensional space we can reduce the dimensionality by performing PCA (Hotelling, 1933) on the global model and setting the target dimensionality to the local dimensionality of the global model (that is the dimensionality of the low-dimensional space).

[3]See the discussion in Section 3.3.3 for more detail on orthogonal projections.

3.1.5 Algorithm Step Through

We now move onto a simple step through of the PLML algorithm so as to provide an example of how the algorithm runs. This step through is shown on the next few pages as Figures 3.7 and 3.8. Throughout this section we drop the Figure identifier and simply refer to the subfigure labels (so, for example, (c) refers to Figure 3.7 (c) and (j) refers to Figure 3.8 (j)). The input data is shown in (a) and consists of a simple 1-dimensional curve in 2-dimensional space. The target dimensionality is selected as 1. The first step is to partition the data using a constrained clustering algorithm (b). Notice that the clusters are created with near equal sizes. The local models are then formed in (c) by projecting the points contained in each cluster onto their associated basis vectors. The minimum spanning tree is then found which defines the model topology (d). A random model is then chosen, (e), to begin the traversal of the MST. The next model is then visited, (f), and the previously visited model is merged onto this current model. Since there are no more new models to visit at this stage of the traversal the global model is back aligned to the previous model (g). The traversal continues, (g-l), until all models are visited. By the time the final model is reached, (l), we have a globally aligned representation of the local models and we can perform PCA on this global model to obtain the low-dimensional embedding (m).

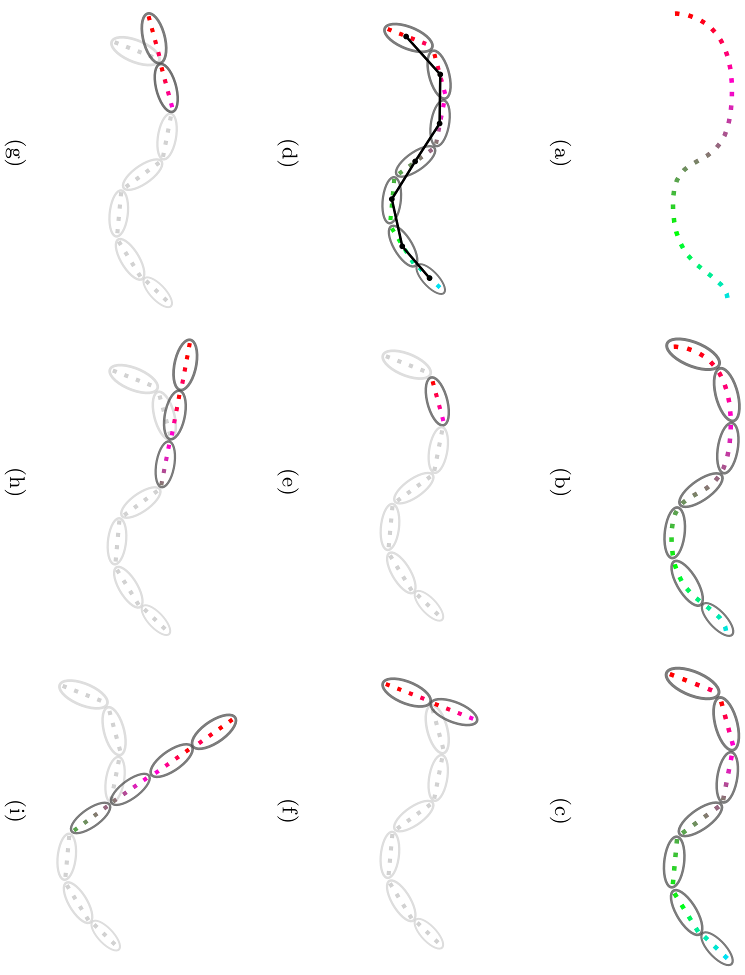
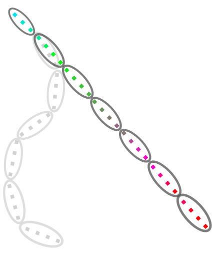
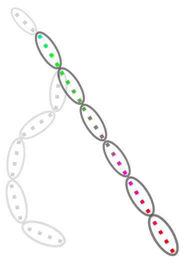
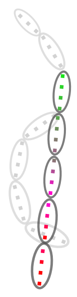


Figure 3.7: A simple example of the steps performed by the Piecewise-Linear Manifold Learning algorithm. Description of this process is found in Section 3.1.5.


(l)


(k)


(j)

(m)

Figure 3.8: Continued from Figure 3.7

3.2 Piecewise-Linear Manifold Learning

In this section we describe the Piecewise-Linear Manifold Learning algorithm in more detail than in the preceding section.

We take as input a high-dimensional set of samples $\mathbf{X} = \{x_i\}_{i=1}^n \in \mathbb{R}^p$ sampled from a low-dimensional manifold $\mathcal{M} \in \mathbb{R}^q$ embedded within $\mathbb{R}^p$ (where $q \ll p$). The goal of any manifold learning algorithm is to recover a set of samples $\mathbf{Y} = \{y_i\}_{i=1}^n \in \mathbb{R}^q$ that best approximate the q-dimensional manifold $\mathcal{M}$. We assume that at a local scale the manifold $\mathcal{M}$ is homeomorphic to Euclidean space and is a C^∞-manifold (i.e. it is smooth differentiable). The two free parameters are the target dimensionality, q, and neighbourhood size parameter, k, which is used to determine the size of the clusters formed in the first step.

3.2.1 Building Local Models via Local PCA

To create the local hyperplanes we first calculate the number of clusters $c = \lfloor n/k \rfloor$ and also a minimum cluster size parameter h. This ensures that, as far as possible, all clusters are created with a similar number of samples. In practice the value of h is set to k since we wish to create as near as possible equal sized clusters. We then use a constrained clustering algorithm to partition $\mathbf{X}$ into c clusters $\{C_l\}_{l=1}^c$ such that the distance between each sample, x_i, and its nearest cluster centre, $\bar{C}_l$ is minimised:

$$\min_{C_1,\ldots,C_c} \sum_{i=1}^{n} \min_{l=1,\ldots,c} \left(\frac{1}{2} \parallel x_i - \bar{C}_l \parallel^2 \right) \tag{3.3}$$

with the specific constraint that no cluster, C_l, is smaller than the minimum cluster size, h, that is $\|C_l\| \geq h$. The above approach is similar to that proposed by Bennett et al. (2000), where a normal k-means clustering algorithm is run with the specific cluster size constraint added. A similar output can be obtained by executing a clustering algorithm and then finding the small clusters (i.e. $|C_l| < h$) and joining them to their closest cluster. This process can reduce the number of clusters to fewer than c, but in the remainder of this section we still refer to c as the number of clusters even though it may be less than the initial value.

Once we have partitioned $\mathbf{X}$ into c clusters we are able to form local models based on local hyperplanes. A local hyperplane relating to the l-th cluster is defined as

$$\Pi_l = \{z \mid z = x\mathbf{U}\mathbf{U}^T,\ x \in C_l\} \tag{3.4}$$

The basis vectors of the hyperplane are then given by the principal components of the samples within C_l and are stored as the $(q+1)$ coordinate vectors in $\mathbf{U}$ found according to $\mathbf{U}\Lambda = \Sigma\mathbf{U}$ and ordered according to their eigenvalues, Λ. The mean point of the hyperplane Π_l corresponds to the cluster centre $\bar{C}_l$.

At this point the data is still embedded within p-dimensional space but is, at a cluster level, q-dimensional. This is due to the fact that the hyperplane projection in Eq. 3.4 is a projection onto the subspace vectors spanned by the data. Although Eq. 3.4 provides us with a definition of a hyperplane the projection of a point, x, to a hyperplane, Π_l, is given by the mapping

$$f : x \to x_l^* = x\mathbf{U}_l\mathbf{U}_l^T + (\bar{C}_l - (\bar{C}_l\mathbf{U}_l\mathbf{U}_l^T)) \tag{3.5}$$

where $x\mathbf{U}_l\mathbf{U}_l^T$ projects x onto the subspace spanned by the column vectors given in $\mathbf{U}$ and $(\bar{C}_l - (\bar{C}_l \mathbf{U}_l \mathbf{U}_l^T))$ is the translation vector that moves $x\mathbf{U}\mathbf{U}^T$ onto the hyperplane Π_l. This translation vector is found by calculating the difference between $\bar{C}_l$ and the image of $\bar{C}_l$ as projected onto the subspace vectors $(\bar{C}_l \mathbf{U}_l \mathbf{U}_l^T)$.

This process warrants a slightly more detailed explanation. It is initially worth noting that $x\mathbf{U}_l\mathbf{U}_l^T$ will project x onto the subspace spanned by the column vectors of $\mathbf{U}$. As such, $x\mathbf{U}_l\mathbf{U}_l^T$ lies on the subspace spanned by the basis vectors of the hyperplane Π_l and therefore we need to find the translation vector that moves it from the origin centered subspace to the hyperplane Π_l. We can do this by projecting a known point on the hyperplane, Π_l, onto the subspace given by $\mathbf{U}\mathbf{U}^T$ and then calculating the translation vector between these two points. This will give us the translation vector needed to move from the subspace to the hyperplane. This is achieved by projecting the mean point of the hyperplane (denoted by the mean point of the cluster, $\bar{C}_l$) onto the subspace (i.e $\bar{C}_l\mathbf{U}_l\mathbf{U}_l^T$). The difference between the two points, $\bar{C}_l$, and, $\bar{C}_l\mathbf{U}_l\mathbf{U}_l^T$, now gives us the translation from the subspace to the hyperplane.

An example of how these models approximate the local structure is shown in Figure 3.9 where we have as input a set of points sampled from an S-curve (Figure 3.9 (a)). The top q-basis vectors of the local hyperplanes are shown in Figure 3.9 (b) and show that the local models lie on the manifold and are a good linear approximation of the local structure.

Since at this stage the data is locally q-dimensional the next step is to align the local models to build a global alignment of the local models. This global alignment will still be embedded in p-dimensional space but will be locally and globally q-dimensional, so a simple PCA transformation will enable us to reduce the dimensionality of the data to q-dimensions. Prior to forming a global model however, we need to determine the topology between the local models.

3.2.2 The Minimum Spanning Tree for Model Topology

The global alignment step of the PLML algorithm is based on the determination of the correct topology between the local models. The topology here refers to the local connectivity of the models, so ideally a model will be connected to its closest neighbours but not to any far away models. As with spectral techniques this neighbourhood graph provides us with a good approximation of the topology between the models and therefore the topology between the data. However, contrary to spectral techniques we do not use this neighbourhood graph as the feature matrix prior to eigendecomposition, rather we use it as a means of traversing the manifold. We wish our topology to be used as a road map describing which other models can be visited from each model. This means that we can traverse the topological graph in a topologically correct order and merge the models as we move from one neighbouring model to another thus forming a continuous global model.

As described in Section 3.1.2 the Minimum Spanning Tree (MST) provides us with a good solution to the above problem as it has many unique and desirable properties. First, as shown by Robins in (Robins, 2000) the MST is ideal as a skeleton for a data set as it tends to avoid shortcuts between branches and it gives a fully connected graph. These two properties overcome the shortcomings of many spectral graph techniques — short circuits (the neighbourhood graph is too large) and disconnected components (the neighbourhood graph is too small). This is shown by the Minimum Spanning Tree in Figure 3.9 (c). However, as discussed in Section 3.1, the MST does not contain enough proximity information for it to be used as a replacement for a k-neighbourhood graph in spectral techniques. The second desirable property of the MST is that it is

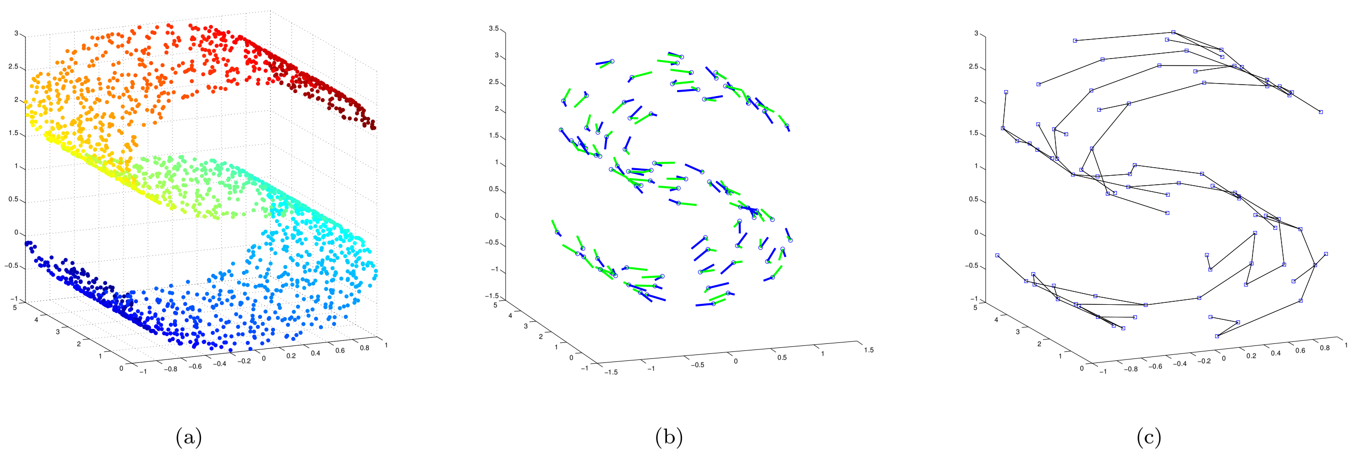


Figure 3.9: The input data (a) and its associated local models as found according to the PLML algorithm (b). The MST of the local models (c) provides us with a good approximation of the connectivity of the models.

non-cyclic. This means that a simple traversal of the MST will not result in the traversal being caught in a local loop but rather all vertices of the MST can be visited in a topological order. It is easy to see therefore how the MST is ideal for our application since we wish to find a way of visiting all nodes in a graph at least once and in a well defined order.

We are now in a position to properly define an MST. A *spanning tree* of a connected undirected graph is a subgraph that connects all the vertices together. If the weights of the graph are given then we can say that the minimum spanning tree of a connected undirected graph is a spanning tree with total weight less than or equal to the weight of every other spanning tree. That is, the Minimum Spanning Tree, T, of a graph, $G = \langle V, E \rangle$, has the same vertex set of G but a reduced edge set $E' \subset E$ (where E' is the edge set of minimal cost). We build the MST of the local models by firstly building an MST of the input data $\mathbf{X}$. If we were to build an MST at model level the cluster centres would provide too sparse an approximation of the manifold to build a suitably robust graph. As such some of the undesirable properties (e.g. short-circuits and disconnected components) would be likely to return.

To build the MST of $\mathbf{X}$ we first calculate the $n \times n$ distance matrix $\mathbf{D}$, where $\mathbf{D}(i,j) = d(x_i, x_j)$ where $d(\cdot)$ is a metric function, in this case returning the Euclidean distance between samples x_i and x_j. We can now build the MST, $T = \langle V, E' \rangle$, of $\mathbf{D}$ where T is a subgraph of $\mathbf{D}$ with the same vertex set V but a reduced edge set $E' \subset E$. This full MST provides us with the basis for our model-wise MST. The model level MST, G, is formed by adding an edge between two models, Π_i and Π_j, if there exists an edge in $T(a,b)$ that connects a point $x_a \in \Pi_i$ and $x_b \in \Pi_j$. A formal definition of our model level MST is given by

$$G(i,j) = \begin{cases} d(\bar{C}_i, \bar{C}_j) & \text{if} \quad \exists e \in E \mid e = \langle v_a, v_b \rangle \\ & \quad\quad v_a \in C_i, v_b \in C_j \\ \infty^+ & \text{otherwise} \end{cases} \tag{3.6}$$

where $d(\bar{C}_i, \bar{C}_j)$ is the Euclidean distance between the cluster centre of C_i and cluster the centre of C_j. The output of the above process is not always guaranteed to produce a Minimum Spanning Tree, so we add the post-processing step of running the MST on G to ensure an MST is created. We now have a graph describing the topology between our models in the form of an MST of the models.

The next step can be seen as the heart of the PLML algorithm: aligning the local models to form a global model.

3.2.3 From Local to Global: Combining Local Models

The core step of the PLML algorithm is that of combining the models formed in Step 1 according to the topology found in Step 2 to produce a global approximation of the manifold. This piecewise combination of the locally linear models is what gives the algorithm its name.

The basic idea is to form a global model by walking along the MST of models and merging the models as the traversal takes place. Figure 3.10 shows the principle of this. Starting at the model Π_2 we wish to move to Π_3 since we know that it is a neighbour of Π_2 through the MST. To move Π_2 to Π_3 we rotate Π_2 so that its axis is aligned with Π_3. We then project Π_2 onto Π_3. Since Π_2 and Π_3 are parallel there is no distortion or change in topology within Π_2 which would not be the case if the two models were not parallel. When moving from Π_3 to Π_4 the steps are repeated but the rotation and projection are applied to Π_2 as well as Π_3 since they are now part of the global merged model. If subsequently we want to incorporate Π_1 then we need to move back from Π_4 to Π_3 (and subsequently Π_2) since Π_3 (and Π_2) have already been aligned and are part of the global model. We do this by aligning Π_3 (and also Π_2) in turn to their original representations. It is also worth noting that we create a global model based on local models that is *separate* from their original representations. Therefore, we denote Π_i as a model as originally created and Π_i' as a model having undergone alignment. So in the last two steps in Figure 3.10 we are aligning Π_3' to Π_3 and subsequently Π_2' to Π_2.

To walk along the MST we use a simple pre-order traversal (Valiente, 2002) which ensures that parents are visited before children, and siblings are visited in left-to-right order. To describe the process of pre-order traversal we denote the first child of a node Π_i as $\mathit{first}[\Pi_i]$. $\mathit{next}[\Pi_i]$ denotes the next sibling of node Π_i, $\mathit{last}[\Pi_i]$ denotes the last child of node Π_i and $\mathit{size}[\Pi_i]$ denotes the number of nodes in the sub-tree of G. $\mathit{order}[\Pi_i]$ gives the order in which Π_i is to be visited. So we visit the first node with $\mathit{order}[\Pi_i] = 1$, then $\mathit{order}[\Pi_j] = 2$, until we reach $\mathit{order}[\Pi_l] = c$. Given a random node, Π_r, set as the root node for traversal, a bijection order $\Psi : V \to \{1, \ldots, c\}$ is a pre-order traversal of G if $\mathit{order}[\Pi_r] = 1$ and $\mathit{order}[\mathit{first}[\Pi_i]] = \mathit{order}[\Pi_i] + 1$ *(if Π_i is not a leaf)*; $\mathit{order}[\mathit{next}[v\Pi_i]] = \mathit{order}[\Pi_i] + \mathit{size}[\Pi_i]$ *(if Π_i is not a last child)*.

As mentioned above, there are two different scenarios that need to be considered when moving from one model to another. If the new model, Π_i, has not been previously visited then we need to project the global model onto the hyperplane defined by this model (**Forward projection**). Otherwise, if the model has been previously visited then we need to align Π_i' to Π_i (**Back aligning**). We deal with each of these cases separately below.

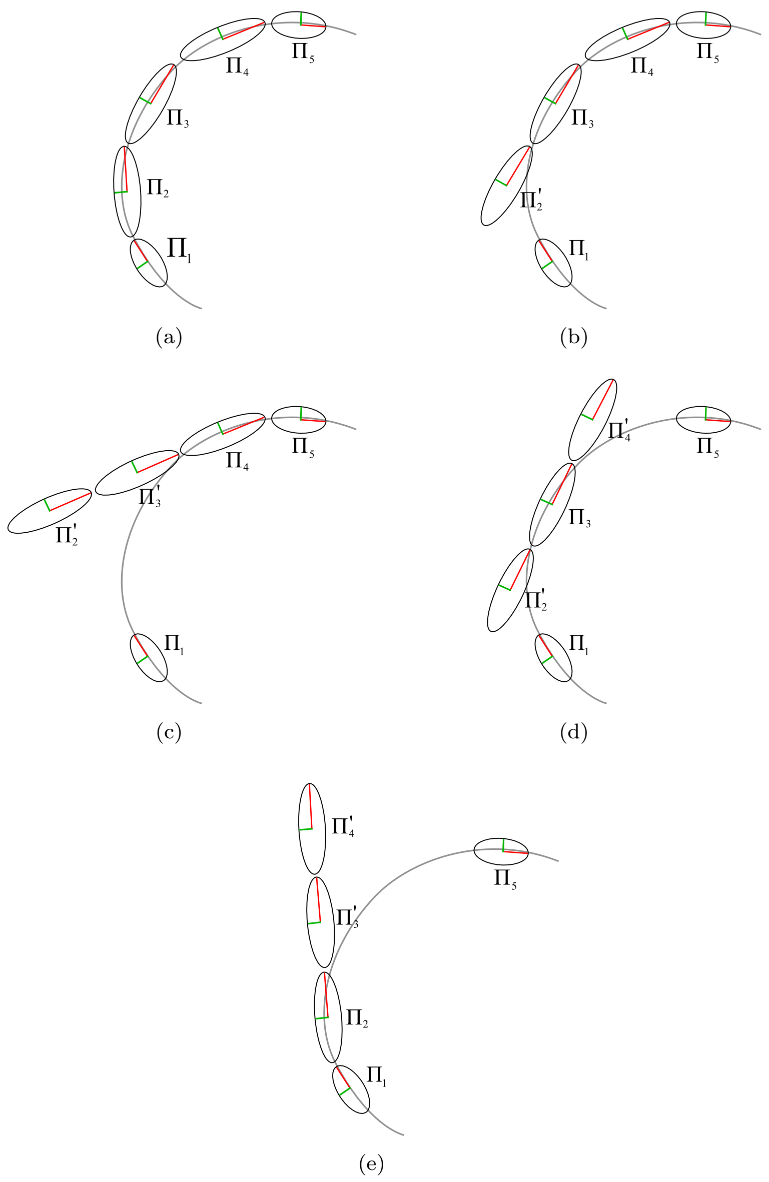


Figure 3.10: An overview of the parallel projection method of the PLML algorithm. The result, shown in (b) is that Π_2 and Π_3 now exist on the same global model. (c) shows how Π_2 and Π_3 move onto Π_4. In (d), the model under alignment, Π_3', is aligned to its original representation Π_3. Similarly, when moving back to Π_2, Π_2' is aligned to Π_2 (e).

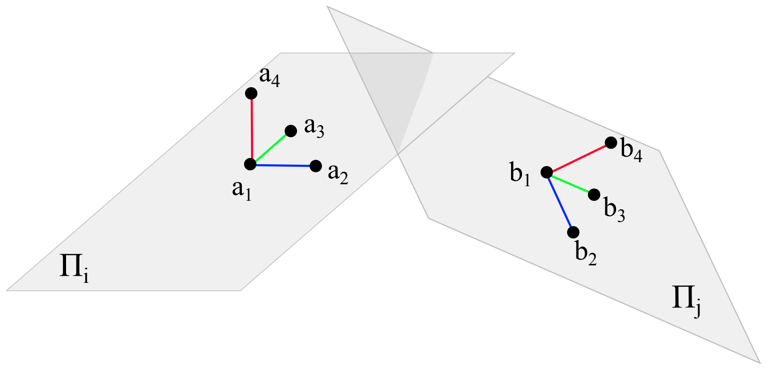


Figure 3.11: Two models Π_i and Π_j represent here by two hyperplanes and their basis vectors. Here a_4 and b_4 are the normal vectors to the two hyperplanes.

Forward Projection

For forward projection we wish to project the previously visited models onto the current model so that they lie on the same hyperplane (i.e. the same co-ordinate space). We know that the previously visited models all lie on the hyperplane spanned by the previous model and we wish to project them onto the hyperplane spanned by the current model. However, this projection process is more involved than a simple orthogonal projection (such as that described in Eq. 3.5) as we are projecting onto basis vectors that are not in the same locally linear model. As such this can introduce distortions into the mappings[4]. To reduce distortions we require the two models to be parallel and it is this parallelisation of projections that is one of the key processes within the PLML algorithm.

The principal insight is that two models are parallel if their normal vectors are aligned[5] . The normal vector to a hyperplane or subspace in higher dimensions is ill defined and so we define a normal vector as the $(q+1)^{\text{th}}$ principal component

[4]See the discussion in Section 3.1.3, in particular Figure 3.6.

[5]Throughout this discussion we interchange between the terms model and hyperplane for ease of reading. Therefore the normal vector of a model is the same as saying the normal vector of the hyperplane spanned by the model.

of the model. This will provide us with a vector that is orthogonal to the basis vectors spanning the model. The alignment of these normal vectors is equivalent to the dihedral angle between the two models being equal to 0. Therefore, the goal becomes finding the rotation needed to align the previous model with the current model such that they are parallel.

The concept of rotations in high-dimensional space is ill-defined, so we cannot simply rotate Π_{prev} by the angular difference between the normal vectors of Π_{prev} and Π_{new} as there are multiple ways of performing this rotation. To explain this further consider the 3-dimensional rotation case. For this we will be given 3 rotation matrices $\mathbf{R}_x$, $\mathbf{R}_y$ and $\mathbf{R}_z$. From these there are 6 different products (ways) of rotation. For 4-dimensional space we no longer rotate around a point but rather rotate around a plane. We now have 6 rotation matrices leading to $6! = 720$ ways of rotating. Furthering this to 5-dimensions it becomes apparent that there will be 10-planes to rotate about leading to $10! = 3628800$ ways of rotation. To overcome this we borrow a concept from point set registration theory (Gower, 1975) to find the rotation matrix between the two models.

This rotation matrix can be found by thinking of the basis vectors of each model as a set of points. In the example shown in Figure 3.11 we have a model Π_i with a set of basis vectors represented by the points a_2, a_3, a_4 and a central point a_1 and a model Π_j with the basis vectors as points b_2, b_3, b_4 and the central point b_1. We wish to find the rotation matrix that aligns Π_j with Π_i (i.e. the angle between the normal vectors $\vec{b_1 b_4}$ and $\vec{a_1 a_4}$ becomes 0). Since we are dealing with two sets of points a simple solution would be to perform Procrustes analysis (Gower, 1975) to find the rotation matrix needed to rotate Π_j to 'fit' Π_i. Procrustes analysis seeks to find the change in translation, rotation and scaling between two sets of points. However, although this will find a rotation matrix to align the two sets of points there is no unique solution as there are multiple ways of aligning these two sets. For example b_4 is not guaranteed to be aligned with a_4, it could be rotated to align with a_3 or a_2. As such we need to mirror the axis and scale each point to a unique size relative to the central point so as to ensure a unique solution. So, for example, the distance $\| a_1 - a_2 \| \neq \| a_1 - a_3 \| \neq \| a_1 - a_4 \|$ but more than that $\| a_1 - a_2 \| = \| b_1 - b_2 \|$. This means that although each axis has been uniquely scaled, the scaling factor for each axis is the same between models. This ensures that there is only 1 solution to the alignment. Figure 3.12 shows this in more detail. We have mirrored each of the axes and scaled them such that the size of the axes is the same for both models. There now exists a unique solution to the alignment problem. That is, b_1 will be aligned with a_1, b_2 will be aligned with a_2, etc.

Constructing a more rigid definition of this process we denote the previously visited model's hyperplane as Π_{prev} and the new model's hyperplane as Π_{new}. We wish to make Π_{prev} parallel to Π_{new} and to do this we align the axis of Π_{prev} to the axis of Π_{new} as described above. Two models, Π_a and Π_b, are defined as parallel if their dihedral angle is 0, that is

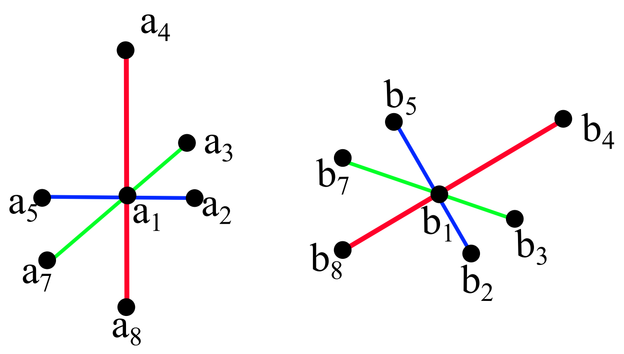


Figure 3.12: Two axis matrices can be aligned by thinking of each as a set of points defining the principal directions of each axis. These axes are mirrored and scaled so as to ensure a unique solution to the alignment problem. In this example b_1 will be aligned with a_1, b_2 with a_2, etc. Although each axis is uniquely scaled relative to each other axis they are still the same size across axes. That is, $\| a_2 - a_1 \|$ is the same size as $\| b_2 - b_1 \|$, etc.

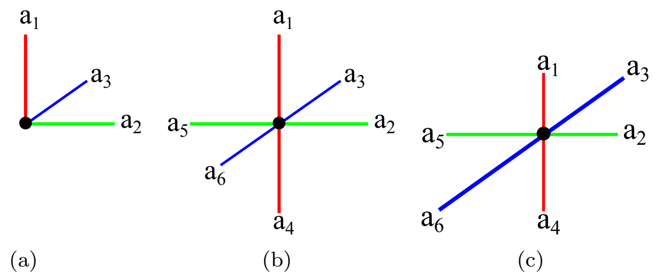


Figure 3.13: The three different type of axis considered. (a) partial, (b) full, (c) full scaled.

$$\varphi(\Pi_a, \Pi_b) = \cos^{-1}(\mathbf{n}_{\Pi_a} \bullet \mathbf{n}_{\Pi_b}) = 0 \tag{3.7}$$

where $\mathbf{n}_{\Pi_a}$ is the normal vector to the model Π_a. So the problem of aligning the axis of Π_{prev} to Π_{new} can be thought of as minimising the dihedral function φ. As we have shown the solution to this problem can be found by constructing a set of points relating to the principal axes of each model. This set of points, referred to as the axis matrix $\mathbf{N}$, will have as columns the top $(q+1)$ eigenvectors and their additive inverse (which creates the 'mirror' of the vectors). The top q eigenvectors are the vectors spanning the low-dimensional hyperplane and the $(q+1)^{\text{th}}$ vector is analogous to the normal to the hyperplane. As such the axis matrix can be thought of as a concatenation of two matrices - the basis vectors $\mathbf{U}_{1\ldots(q+1)}$ and their additive inverse ${}^{-}\mathbf{U}_{1\ldots(q+1)}$. So we can formally define an axis matrix as:

$$\mathbf{N} = [\mathbf{U}_{1\ldots(q+1)} \; {}^{-}\mathbf{U}_{1\ldots(q+1)}] \tag{3.8}$$

Figure 3.13 (b) shows pictorially the properties of this axis matrix. Although we have successfully mirrored the axis we have not scaled them and as such there is still no unique solution to the alignment problem. To ensure a unique solution we need to scale each of the axes.

To scale the axis matrix we first create a $p \times (q+1)$ scale matrix $\mathbf{S}$. This scale matrix is defined as:

$$\mathbf{S}(i,j) = \begin{cases} i & \text{if}(i \leq (q+1)) \\ i + (\frac{1}{2}) & \text{otherwise} \end{cases} \tag{3.9}$$

such that the scale is unique for each column (axis) but repeated along each row. According to Eq. 3.9 the scale matrix for 3-dimensional space would be

$$\mathbf{S} = \begin{bmatrix} 1 & 2 & 3 & 1.5 & 2.5 & 3.5 \\ 1 & 2 & 3 & 1.5 & 2.5 & 3.5 \\ 1 & 2 & 3 & 1.5 & 2.5 & 3.5 \end{bmatrix} \tag{3.10}$$

To apply the scaling matrix we take the Hadamard product[6] of the two matrices such that we can now define a full axis matrix as:

$$\mathbf{M} = \mathbf{N} \circ \mathbf{S} \tag{3.11}$$

The result of applying this scale matrix is shown in Figure 3.13 (c).

Now that the axis matrices have been defined we can return to the point cloud registration problem. Given the two axis matrices $\mathbf{M}_{new}$ and $\mathbf{M}_{prev}$ we wish to find the rotation matrix $\mathbf{R}$ so that $\| \mathbf{M}_{new} - \mathbf{M}_{prev}\mathbf{R} \|$ is minimised (where $\| \cdot \|$ is the Frobenius norm). The matrix $\mathbf{R}$ is orthonormal and gives the rotation needed to align the axis matrix $\mathbf{M}_{prev}$ to $\mathbf{M}_{new}$. Wang and Mahadevan

[6]The Hadamard product is the element wise multiplication of two same sized matrices, $\mathbf{A}$, $\mathbf{B}$, such that $(\mathbf{A} \circ \mathbf{B})_{ij} = \mathbf{A}_{ij} \cdot \mathbf{B}_{ij}$

(2008) have shown that the optimal solution is given by the Singular Value Decomposition (SVD) of $\mathbf{M}_{prev}^T\mathbf{M}_{new}$. That is, if the SVD of $\mathbf{M}_{prev}^T\mathbf{M}_{new}$ is $\mathbf{U}\Sigma\mathbf{V}^T$ then $\mathbf{R} = \mathbf{U}\mathbf{V}^T$.

To apply rotation we simply perform the matrix multiplication $\Pi'_{prev} = \Pi_{prev}\mathbf{R}$ which can be thought of as minimising Eq. 3.7. Now that the two models are parallel we project Π'_{prev} onto Π_{new} as follows:

$$\Pi'_{\text{prev'}} = \Pi'_{\text{prev}}\mathbf{U}_{\Pi_{\text{new}}}\mathbf{U}_{\Pi_{\text{new}}}^T + (\Pi_{\text{new}}^{-} - (\Pi_{\text{new}}^{-}\mathbf{U}_{\Pi_{\text{new}}}\mathbf{U}_{\Pi_{\text{new}}}^T)) \tag{3.12}$$

where $\mathbf{U}_{\Pi_{\text{new}}}$ is a matrix containing the top q principal components of Π_{new}. Since Π_{prev} is the global model, this rotation and project will rotate and project all the previously visited models onto the current model.

Back Aligning

If the model we wish to move to has already been visited then we do not need to worry about the parallel projection process described above. Rather we simply wish to align the model to its original representation. If we wish to align Π'_i to Π_i then we need to find the translation vector that moves the centroid of Π'_i onto the centroid of Π_i and also the rotation matrix that correctly aligns the samples in Π'_i to Π_i. This step can easily be achieved by running Procrustes analysis on the two sets of samples within the two models. Since the number of samples within the models will not have changed and the forward projection step described above does not distort the samples within the model we can use Procrustes analysis (Gower, 1975; Sibson, 1978; Cox and Cox, 2001) to find the translation vector v and the rotation matrix $\mathbf{R}$ to match Π'_i to Π_i. The translation v and rotation matrix $\mathbf{R}$ can be found as follows. Let $\mathbf{Z} = \Pi_i^T\Pi_j$

and we denote its Singular Value Decomposition (SVD) as $\mathbf{Z} = \mathbf{ULV}^T$ (Mardia et al., 1979). The Procrustes rotation matrix is given by $\mathbf{R} = \mathbf{UV}^T$ and the translation vector is given by $v = \bar{\Pi}_i - \mathbf{R}\bar{\Pi}'_i$ (where $\bar{\Pi}$ represents the mean point of the model) (Sibson, 1978). Therefore when back aligning, $\Pi'_i = \Pi'_i\mathbf{R} + v$. As with the forward projection step this translation and rotation is applied to the global model rather than just the previously visited model.

3.2.4 Obtaining Low-Dimensional Representation

Once all nodes have been visited in the traversal described above, that is all nodes have been visited and in the correct order given by Ψ (see Section 3.2.3), we will have obtained a globally aligned q-dimensional representation of the manifold embedded within p-dimensional space. This representation is contained in the models $\{\Pi'_l\}_{l=1}^c$. We set the matrix $\mathbf{Q} = \{\Pi'_l\}_{l=1}^c$ to contain all the samples of the globally aligned representation. To find the low-dimensional embedding we run PCA on $\mathbf{Q}$ such that

$$\Lambda\mathbf{V} = \mathbf{C_Q}\mathbf{V} \tag{3.13}$$

where $\mathbf{C_Q}$ is the covariance matrix of $\mathbf{Q}$ and $\mathbf{V}$ is a matrix containing as columns the top q-dimensional eigenvectors sorted according to their associated eigenvalues, Λ. The low-dimensional representation $\mathbf{Y}$ is then given by $\mathbf{Y} = \mathbf{QV}$.

3.3 Alternative Paths Investigated

The purpose of this section is to outline the alternative paths of research that were investigated during the development of the PLML algorithm but did not explicitly contribute to the final algorithm described above. These alternative

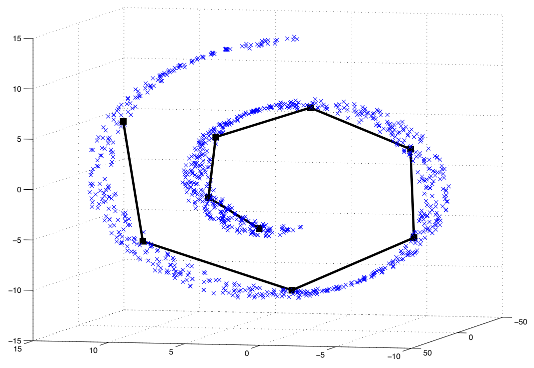


Figure 3.14: The Swiss Roll dataset topology as determined by the GMM clustering scheme. Notice how simple the topology graph is. This makes it significantly easier for the PLML to execute as it is essential a 1-dimensional embedding problem. However, this type of structure is rarely captured by GMM.

paths represent interesting problems that could be used as a springboard for future improvements and research.

3.3.1 Alternative Clustering Techniques

Before a constrained k-means clustering approach was employed to form the local models in Step 1 of the PLML algorithm we used a Gaussian Mixture Modelling (GMM) approach (Bouman, 1997; Sahbi, 2008). This approach was similar to that used in Manifold Charting (Brand, 2003). A mixture of Gaussians was trained over the data set but rather than leading to a soft partitioning of the data as in Manifold Charting we used the GMM to cluster (hard partition) the data. In some cases this use of a GMM scheme led to a more meaningful clustering of the data. For example, the Swiss Roll data set is clustered in such a manner that the models are connected according to a 1-dimensional topology graph (Figure 3.14). This, however, turned out to be the exception rather than the rule. Most other data sets were clustered in much the same manner as

k-means.

The main draw back of the GMM approach was computational complexity. For a large high-dimensional dataset it proved impossible to obtain a clustering on the hardware we had available. This meant that it was not suitable for large scale applications which was one of the attributes we wanted the PLML algorithm to have.

Another clustering scheme investigated was MST Clustering (Grygorash et al., 2006). Although this reduced the complexity of the algorithm (as the MST was already being used to determine the model topology) the quality of the local models proved to be poor. One of the strengths of the k-means approach is that it creates as close to 'spherical' clusters as possible which is ideal for the formation of local linear hyperplanes. There was however large variation in the size and shape of the clusters produced using the MST clustering approach. This in turn led to incorrect or unusable local hyperplane information leading to a heavily distorted or completely unstable low-dimensional embedding.

3.3.2 Intersecting Hyperplanes for Model Topology

Before investigating the use of the Minimum Spanning Tree as a way of modelling the topology between local models we exploited hyperplane-hyperplane intersection as a method for determining the model topology. This method actually presents a new approach to manifold learning and is heavily based on Isomap (Tenenbaum et al., 2000) with the major difference being the construction of the neighbourhood graph. Rather than using Euclidean distance to measure the distance between hyperplanes we use a novel distance - the hyperplane-hyperplane distance. Once we have determined the low-dimensional positions of each hyperplane we can find the global alignment of the hyperplanes by simply translating the low-dimensional representations as found in Step 1 of the PLML algorithm to their position as determined by low-dimensional topology (Figure 3.15)

The hyperplane-hyperplane distance can be broadly defined as the sum of distances between the mean of each hyperplane and these mean points as projected onto their intersecting hyperplane. The key insight here is that the intersection of two q-dimensional hyperplanes is a $(q-1)$-dimensional hyperplane. The distance between a hyperplane and this $(q-1)$-dimensional hyperplane can be found by measuring the difference between a known point on the hyperplane and its image as projected onto the intersecting hyperplane. To perform this projection we need to find a point on the intersecting hyperplane and also its basis vectors.

To find the intersection point between two hyperplanes we firstly find a point on the hyperplane and then determine its basis vectors. Once we have a point on the hyperplane and the hyperplane's basis vectors we can project points onto this hyperplane according to Eq. 3.5. Given two hyperplanes Π_i and Π_j, a

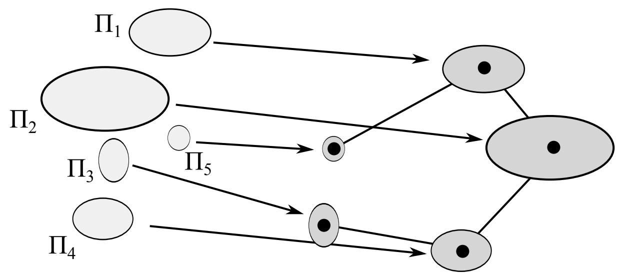


Figure 3.15: An example of how the intersecting hyperplanes methods acquires a global alignment of the local models. The model topology is embedded according to the hyperplane-intersection distance and Isomap (Tenenbaum et al., 2000) as shown by the graph on the right. The local models, Π_1 to Π_5, are then translated so that their mean points align with the associated mean points in the model topology.

point, α, can be found on the intersecting hyperplane $\Pi_k = \Pi_i \cap \Pi_j$ by solving the linear system

$$\begin{bmatrix} \mathbf{n}_i \\ \mathbf{n}_j \end{bmatrix} \alpha = \begin{bmatrix} \mathbf{n}_i \cdot |\Pi_i| \\ \mathbf{n}_j \cdot |\Pi_j| \end{bmatrix} \tag{3.14}$$

where $\mathbf{n}_i$ and $\mathbf{n}_j$ are the normal vectors to the hyperplanes which can be represented by the $(q+1)^{\text{th}}$ principal component of the hyperplane[7].

Now that we have a point on the intersecting hyperplane we need to find the basis vectors of this hyperplane so as to be able to project onto it. This is performed by finding the intersecting subspace between the two hyperplanes. Considering the basis vectors of the two hyperplanes, Π_i and Π_j, as two subspaces we know that the intersection of these two subspaces will be analogous to the basis vectors of the intersecting hyperplane. We denote $\mathbf{U}_i$ as the q-dimensional subspace spanned by the hyperplane of Π_i and $\mathbf{U}_j$ as the q-dimensional subspace spanned by the hyperplane of Π_j. Since any pair of subspaces are proper subspaces their intersection $\mathbf{U}_i \cap \mathbf{U}_j$ is also a subspace. If $\mathbf{U}_i \in \mathbb{R}^p$ and $\mathbf{U}_j \in \mathbb{R}^p$ then $\mathbf{U}_i \cap \mathbf{U}_j$ is also a subspace of $\mathbb{R}^p$ (Grossman, 1994).

To find an explicit basis for the subspace $\mathbf{U}_i \cap \mathbf{U}_j$ we follow the method outlined by Yang in (Yang, 1997). Given the set of q basis vectors of $\mathbf{U}_i$ and $\mathbf{U}_j$ we concatenate them to form the $p \times 2q$ matrix $\mathbf{A} = [\mathbf{U}_i \mathbf{U}_j]$. We then reduce $\mathbf{A}$ to its reduced row echelon form rref($\mathbf{A}$). As shown in (Yang, 1997) the t linear combinations $a_1u_1 + a_2u_2 + \ldots + a_qu_q$ form a basis for the intersection represented as $\mathbf{V}$ (where t is the number of non-pivotal columns in $\mathbf{A}$).

[7]An alternative method of finding the normal vector is to take the generalised cross product of the q basis vectors (Hanson, 1994). This is because in 3-dimensional space the normal vector to the plane can be found by taking the cross product of the plane's basis vectors. However in n-dimensional space this generalised cross product becomes difficult to calculate as the generalisation is based on finding the cofactors of the determinant of a large dense square matrix (Hanson, 1994)

Since we now have a point on the intersecting hyperplane, α, and the basis vectors of this intersecting hyperplane, $\mathbf{V}$, we can now define the hyperplane intersection distance. Given two hyperplanes, Π_i and Π_j, and their sample mean points, π_i and π_j, we represent the hyperplane distance, h, between Π_i and Π_j as

$$h(\Pi_i, \Pi_j) \; = \parallel \pi_i - (\pi_i \mathbf{V}\mathbf{V}^T - (\alpha - (\alpha \mathbf{V}\mathbf{V}^T))) \parallel_2^2 \tag{3.15}$$

Since we wish to find the bi-directional distance between Π_i and Π_j we sum the two hyperplane distances so the total hyperplane distance H becomes

$$H = h(\Pi_i, \Pi_j) + h(\Pi_j, \Pi_i) \tag{3.16}$$

Once we have determined the hyperplane-hyperplane distance between each model we can then take the k-smallest distances as the k model neighbours. Once we have formed this hyperplane level neighbourhood graph we follow the same steps as Isomap (Tenenbaum et al., 2000) to reduce the dimensionality of the model topology. With the model topology embedded in q-dimensional space we find the global low-dimensional embedding by aligning the mean of the q-dimensional representation of each model with the mean as found by performing Isomap on the model topology.

Although some results were obtained on artificial data using this approach (an example embedding of the Swiss Roll dataset is shown in Figure 3.16) there were many drawbacks which left it as an unfeasible option. First, there were no significant improvements in performance over Isomap. Since this algorithm was based so heavily on the Isomap algorithm it would be expected to outperform Isomap if it were to be seen as an improvement over existing approaches. As it did not outperform Isomap this raised questions as to its suitability. As well as this the computational complexity was very high. At the heart of the algorithm is the need to solve a large linear system (Eq. 3.14) and also perform the computationally expensive task of reducing a large matrix to its reduced row echelon form. This meant that we were unable to gather results for any data set with dimensionality larger than 3. Finally, the method of reducing the model topology and then aligning the local models to determine the global alignment was fundamentally flawed. The flaw lies in the fact that Isomap does not guarantee to preserve the topology and local structure of the data. As such inter-model distances will change between the high and low-dimensional spaces and so distortions and overlaps will occur when aligning the local models to the reduced topology graph.

So although this method provided a novel approach to determining the connectedness of the models it was not robust enough to produce any meaningful

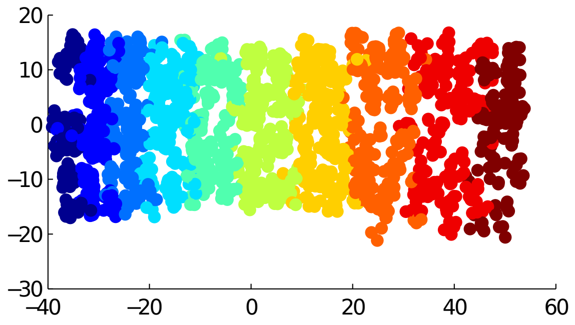


Figure 3.16: An embedding of the Swiss Roll dataset as found by the intersecting hyperplane method. Notice that even though the data has zero noise there is still apparent noise in the embedding. This is due to the fact that the low-dimensional embedding can become distorted by changes in scale to the topology graph.

results.

3.3.3 Non-Parallel Projections

At the heart of the PLML algorithm is the parallel projection step (Section 3.1.3). However the first iteration of the PLML algorithm did not include this parallel projection step (Strange and Zwiggelaar, 2010). Instead it relied solely on the orthogonal projection between models. If the dihedral angle between models is small then the amount of distortion introduced by orthogonal projections is minimal. However, once the dihedral angle increases the distortions become more obvious and problematic

3.4 Conclusions

In this chapter we have presented the PLML algorithm for manifold learning. The algorithm works by modelling the manifold as a set of discrete locally linear models that are locally low-dimensional. The topology between these models can be represented by a Minimum Spanning Tree which provides us with an ordering with which we can traverse these models. By visiting these models in a topologically correct order we can build a global representation of these local models. This global representation is built by merging the models at each traversal step. Once all the models have been visited we will have built a global alignment of the local models.

The next chapter is concerned with providing a detailed analysis of the performance of the PLML algorithm.

“Quapropter bono christiano, sive mathematici,
sive quilibet impie divinantium,
maxime dicentes vera, cavendi sunt,
ne consortio daemoniorum irretiant”

Augustine of Hippo (354 - 430)

4

Piecewise-Linear Manifold Learning: Performance and Analysis[1]

In this chapter we perform a rigorous analysis of the PLML algorithm testing it on both artificially generated data and real (image) datasets. These two different categories allow us to draw conclusions as to how the PLML algorithm performs. The artificially generated datasets provide us with an environment within which we can control the structure and properties of the data. Using this we can change the density and noise levels of the datasets and thereby see how PLML performs with sparse and noisy data. We can also use the artificially generated data to explain how PLML performs on manifolds which contain essential loops or discontinuities. The image data on the other hand allows us to test the performance of PLML on real world data. These datasets often lie

[1]Harry Strange and Reyer Zwiggelaar. Parallel Projections for Manifold Learning. In Proceedings of the ICMLA, Washington DC. IEEE Press, December 2010.

on difficult and non-trivial manifolds but the embedding produced by PLML should allow us to make inferences as to the structure of the data.

The PLML algorithm is considered side-by-side with existing state of the art manifold learning algorithms. These algorithms not only represent the benchmark performance but also the different paradigms used for manifold learning. As such one of the purposes of this chapter is to see how well PLML performs when compared with existing, previously evaluated, manifold learning algorithms.

In this chapter we also consider the computational complexity of the PLML algorithm and compare it with the computational complexities of some other leading manifold learning algorithms.

The remainder of this chapter is structured as follows. We begin in Section 4.1 by outlining the methodology we will use in the experiments in this chapter. We discuss the quality measures we use, the other manifold learning algorithms we compare against, and also the environment within which we perform the experiments. We begin our analysis in Section 4.2 by showing visually how our algorithm fairs against the comparison algorithms on four artificial datasets. In Section 4.3 we perform a detailed analysis on the Swiss Roll dataset. We compare against four state of the art manifold learning algorithms and investigate the effects of sparsity, noise and parameter selection. In Section 4.4 we show how the PLML algorithm performs on image data and then we discuss the computational complexity and memory requirements in Section 4.5. We end with Section 4.6 by drawing the conclusions obtained from the analysis in this chapter.

4.1 Methodology

In this section we outline the methodology used during the analysis of the PLML algorithm. We begin by visually assessing the quality of embeddings produced by PLML and four other manifold learning algorithms on four artificially generated datasets. This visual assessment allows us to gain a high-level understanding as to how the PLML algorithm performs on different types of manifold. The visual comparison also allows us to see how the performance of PLML differs from existing manifold learning techniques and allows us to identify the properties of the low-dimensional embeddings produced by PLML.

We then move on to analyse the performance of PLML in a detailed and quantitative manner. By focusing solely on the Swiss Roll dataset we are able to assess how the PLML algorithm copes with a non-linear dataset under different densities, noise levels and parameters. We compare PLML's performance with the four existing manifold learning algorithms so as to gain a better understanding of the strengths and weaknesses of PLML as well as the differences and similarities it shares with certain techniques. For this detailed analysis we use three different quality measures which measure the stability of a low-dimensional embedding at local and global scales.

Our assessment of the PLML algorithm is then rounded off by applying it to three different image datasets. These datasets exhibit difficult manifold structure and we investigate the results obtained by PLML. This qualitative analysis of the image datasets gives us insight into how the PLML algorithm works on real world data.

4.1.1 Quality Measures

The purpose of the quality measures is to assess the stability of a manifold learning algorithm's embedding of a given dataset. Therefore the choice of appropriate quality measures is essential to gain a solid understanding of how an algorithm performs. The correct choice of quality measure was listed as an open problem in Chapter 2 and as such there is no gold standard. Instead we use three different quality measures to assess the performance of a manifold learning algorithm at both a local and global scale.

Venna rightly postulated that the retention of local properties of a dataset are often of more importance than the retention of global properties (Venna and Kaski, 2006). As such two of the three quality measures that we employ are concerned with the local stability of the embedding, one measuring the topological stability (Trustworthiness), and one measuring the geometric stability (Procrustes Error). The third quality measure we use is concerned with how well the global distances, and therefore global structure, of the data has been maintained (RMSE).

Trustworthiness

Trustworthiness (Venna and Kaski, 2006) measures the retention of neighbours between the high and low-dimensional spaces. An embedding achieves a high Trustworthiness value if the set of k nearest neighbours of a point in $\mathbf{Y}$ are also nearest neighbours of $\mathbf{X}$. As such Trustworthiness is a good measure of how well the local neighbourhoods have been reconstructed in the low-dimensional space.

We follow the same terminology and methodology as in (Venna and Kaski, 2006). The Trustworthiness of an embedding over a neighbourhood size region

k is defined as

$$T_k = 1 - \frac{2}{nk(2n-3k-1)} \sum_{i=1}^{n} \sum_{j \in U_k(i)} (r(x_i, x_j) - k) \tag{4.1}$$

where U_k is the set of data samples that are in the k-neighbourhood of x_i in the low-dimensional space but are not neighbours in the high-dimensional space. $r(x_i, x_j)$ is the rank of the data sample x_j in the ordering according to the distance to x_i in the high-dimensional space. So a large value of $r(x_i, x_j)$ will occur if the point x_j enters into the neighbourhood of x_i from a long distance in the high-dimensional space. The front fraction, $\frac{2}{nk(2n-3k-1)}$, is a normalising term. A Trustworthiness value of 1 indicates that no local neighbourhood distortions have occurred as a result of manifold learning.

Procrustes Error Measure

The Procrustes Error Measure (Goldberg and Ritov, 2009) measures the local distortion introduced as a result of manifold learning. Since this error metric is based on Procrustes analysis it measures the change in rotation between a neighbourhood in high-dimensional space and that same neighbourhood in the low-dimensional space. The error measure is defined as

$$R_k(\mathbf{X}, \mathbf{Y}) = \frac{1}{n} \sum_{i=1}^{n} G(x_i, y_i) \tag{4.2}$$

where $G(x_i, y_i)$ returns the Procrustes statistic of the neighbourhood around x_i and y_i and is measured as follows

$$G(x_i, y_i) = \inf_{\{\mathbf{A}, a : \mathbf{A}'\mathbf{A} = \mathbf{I}, b \in \mathbb{R}^q\}} \sum_{i=1}^{k} \parallel x_i - \mathbf{A} y_i - n \parallel^2 \tag{4.3}$$

$$= \inf_{\{\mathbf{A}, a : \mathbf{A}'\mathbf{A} = \mathbf{I}, b \in \mathbb{R}^q\}} \operatorname{tr}((\mathbf{X} - \mathbf{Y}\mathbf{A}' - 1b')'(\mathbf{X} - \mathbf{Y}\mathbf{A}' - 1b')) \tag{4.4}$$

where $\mathbf{A}$ is the Procrustes rotation matrix and b is the Procrustes translation vector[2].

A Procrustes Error value of 0 indicates no distortion between a given neighbourhood in the high and low-dimensional spaces. The function R_k is sensitive to scaling and so will penalise those techniques that normalise the data or introduce local scale changes.

One question which arises out of the use of the Procrustes error measure is whether it is biased towards favouring PLML as the PLML algorithm uses Procrustes algorithm to align the local models during Back Alignment (Section 3.2.3). Although PLML does use the Procrustes algorithm it does not find the low-dimensional embedding my minimising the Procrustes error measure (Equation 4.2). If it did, then the above error measure would be biased towards the PLML algorithm as it is this measure which should be minimised during execution of the algorithm. However, since PLML only uses Procrustes to align previously moved models, the Procrustes error measure is never explicitly minimised. This is in contrast to the Greedy Procrustes and Simulated Annealing presented by Goldberg and Ritov (2009) where Equation 4.2 is minimised to obtain the low-dimensional embedding.

[2]For computation of these values see either Chapter 3 or (Goldberg and Ritov, 2009; Mardia et al., 1979; Sibson, 1978)

Residual Mean Square Error

The Residual Mean Square Error (RMSE) sometimes called the Residual Variance measures the difference in distances between all points in the high and low-dimensional space (Tenenbaum et al., 2000). The error is measured by

$$\rho_k = 1 - R^2(\mathbf{D_X}, \mathbf{D_Y}) \tag{4.5}$$

where $\mathbf{D_X}$ is a square symmetric matrix containing the distances between all points in the high-dimensional space (as measured by the geodesic distance graph formed by the k-nearest neighbours). $\mathbf{D_Y}$ is the Euclidean distance matrix between points in the low-dimensional space. R^2 is the linear correlation coefficient, taken over all entries of $\mathbf{D_X}$ and $\mathbf{D_Y}$.

RMSE provides us with a good approximation of how well the manifold learning technique has preserved the distances during the manifold learning process. A value of 0 would indicate a perfect correlation between the two distance matrices.

4.1.2 Comparison Algorithms

For an in depth analysis we compare the PLML algorithm against 4 leading manifold learning algorithms. These algorithms are discussed in more detail in Chapter 2 and have been selected because each represents a different paradigm in manifold learning and are the leading algorithms within each paradigm.

Principal Components Analysis (PCA) (Hotelling, 1933) is possibly the most widely used dimensionality reduction technique and represents not only linear dimensionality reduction but also the *goto* method that many people use when considering manifold learning. Because of this PCA can be seen as the baseline method.

Isomap (ISO) (Tenenbaum et al., 2000) is a global approach to manifold learning. It uses geodesic distances to calculate the inter-point distance matrix and is one of the most successful and well known manifold learning algorithms.

Locally Linear Embeddings (LLE) (Roweis and Saul, 2000) is a local approach that appeared at the same time as Isomap. It works by calculating the local linear reconstruction weights to reproduce a point in terms of its nearest neighbours.

Local Tangent Space Alignment (LTSA) (Zhang and Zha, 2004) is a global alignment of local linear models technique that uses the alignment of local tangent spaces to build a global model of the manifold. Since LTSA is a global alignment of local models technique it can be seen as PLML's closest relative.

4.1.3 Environment

All experiments were performed in the MATLAB programming environment[3]. The implementations for PCA, Isomap, LLE and LTSA were taken from Laurens Van Der Maaten's Dimensionality Reduction toolbox for MATLAB[4]. The experiments were performed on both a Dell Optiplex 755 with a 2.4GHz Intel Dual CPU with 3.25GB of RAM running Windows XP Professional, and a MacBook Pro with a 2.7GHz Intel Core i7 and 8GB of RAM running Mac OS 10.6.7.

[3]MATLAB: `http://www.mathworks.com` (*Link checked 26/07/11*)

[4]The toolbox is available from `http://bit.ly/9qtylr` (*Link checked 26/07/2011*).

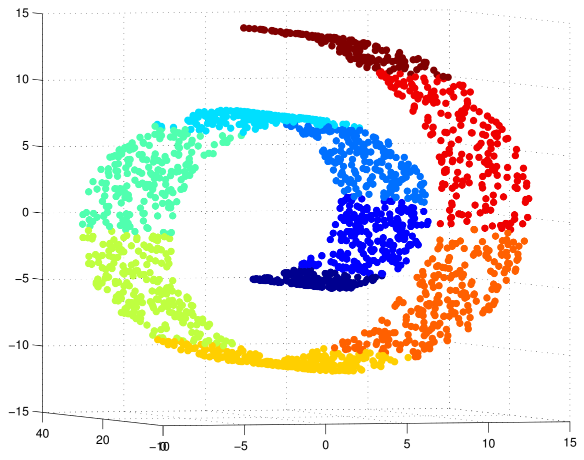
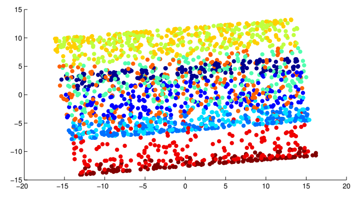


Figure 4.1: Two thousand (2000) points sampled from the Swiss Roll dataset.

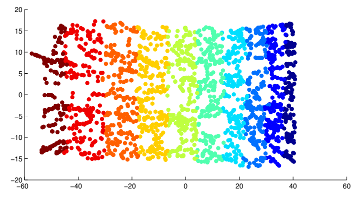
4.2 Method Comparison

In this section we provide an overview of the different approaches to manifold learning on toy manifolds. The simplicity of the manifolds provide us with a good way of visualising how the different approaches behave when reducing the dimensionality of a dataset.

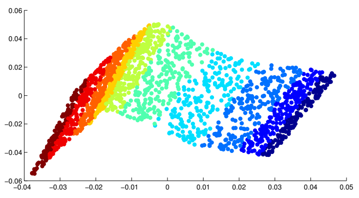
We compare the four algorithms described in the previous section with the PLML algorithm onfour different datasets: the Swiss Roll, the broken Swiss Roll, the Fishbowl and the Helix. For each dataset the optimal parameter was selected for each algorithm. This varies between datasets and was chosen by eye-balling the results and choosing the most visually accurate and pleasing embeddings.



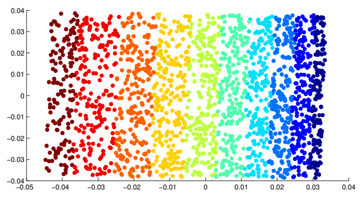
(a) PCA



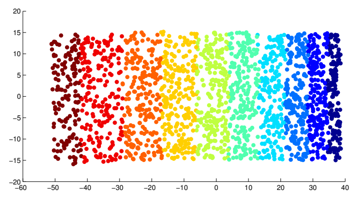
(b) Isomap



(c) LLE



(d) LTSA



(e) PLML

Figure 4.2: The 2-dimensional embeddings of the Swiss Roll dataset.

4.2.1 Swiss Roll

The Swiss Roll is the benchmark dataset used in manifold learning[5]. It contains many properties which make it a difficult and interesting manifold to learn. Simplistically it can be thought of as a spiral with an added 'depth' dimension. An example of the Swiss Roll dataset is shown in Figure 4.1. Here 2000 points are sampled from the Swiss Roll data with zero added noise. As can be seen the manifold exhibits a highly curved structure and can be represented by a 2-dimensional plane.

[5]The dataset originates from evaluating the capabilities of Isomap (Tenenbaum et al., 2000).

The Swiss Roll was initially used to show the benefits of graph based techniques (Tenenbaum et al., 2000). The developable nature of the dataset means that those methods that represent the distances in terms of graph distances will be able to successfully unroll the data. On the other hand, those methods that use Euclidean distances will fail to find a stable embedding of the dataset. This is shown by the embeddings in Figure 4.2. PCA, which uses Euclidean distances, fails to uncover the true manifold structure of the data, whereas Isomap successfully unrolls the manifold. This is not surprising as the data exhibits real non-linear structure and PCA by nature is unable to deal well with non-linear datasets.

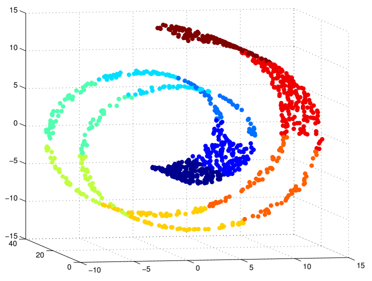
What is of real interest is the comparison of the 4 non-linear techniques. Of these the 2 that produce the most visually appealing embeddings are LTSA and PLML. Isomap unrolls the manifold and recovers a good global reconstruction of the data, however the local neighbourhoods become heavily distorted. This is shown by the expansion effect seen at a local level in Figure 4.2 (b). Some of the data points seem to be pushed outwards and there are holes within the embedding. Conversely, LLE recovers the local neighbourhood structures well but fails to obtain a good global reconstruction of the data. Figure 4.2 (c) shows the LLE embedding. Although locally the structure of the manifold is retained, globally it appears heavily distorted. These results are in keeping with the nature of the algorithms. Isomap is a global technique which can deal well with the global structure of the data but often fails at recovering the local properties. LLE can recover the local structure well but often fails at the global scale.

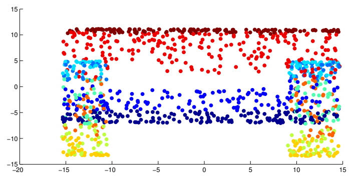
LTSA and PLML are the two techniques that seek to overcome these limitations by globally aligning local linear models. The efficacy of this approach to manifold learning is shown by the results obtained in Figure 4.2. Both LTSA and PLML recover visually pleasing, locally accurate and globally stable embeddings of the dataset. At first it is difficult to separate the two results, however on closer inspection it becomes apparent that the PLML embedding is an improvement over the LTSA embedding. This is due to the normalisation of the LTSA embedding. One of the artefacts of LTSA is that the final embedding is normalised. This normalisation is often far from ideal and can lead to problems not only in the embedding but also further down the processing pipeline (a problem discussed in detail in (Goldberg et al., 2008)). PLML on the other hand does not normalise the data and distances in the low-dimensional embedding are almost identical to those in the high-dimensional space.

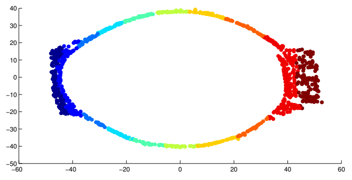
Another point to note at this stage is that although the data is near optimal (i.e. it is noise free and well sampled) some algorithms fail to produce optimal embeddings. Both Isomap and LLE fail to recover either the true local or the true global structure of the data. PCA obviously cannot successfully embed the manifold and LTSA, although it produces a visually appealing embedding, normalises the output. PLML produces the best embedding under these conditions.

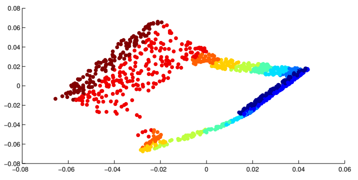
4.2.2 Broken Swiss Roll

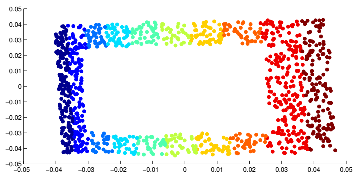
An interesting variant of the Swiss Roll is the broken Swiss Roll dataset. This data is the same as the normal Swiss Roll dataset but with a rectangular area punched out of the middle (Figure 4.3 (a)). The presence of a discontinuity in the manifold has drastic effects on graph based techniques. This, as pointed out in (Lee and Verleysen, 2007), is due to the fact that the graph based distances when considering inter-point distances across the hole are not equal to Euclidean distance across the manifold. This explains the stretched-hole effect of the Isomap embedding as shown in Figure 4.3 (c). LLE also struggles to produce a meaningful global embedding. In fact the embedding quality is decreased as


(a) Broken Swiss


(b) PCA


(c) Isomap


(d) LLE


(e) LTSA

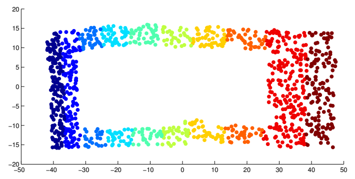

(f) PLML

Figure 4.3: The Broken Swiss Roll (a) and the 2-dimensional embeddings of the (b-f).

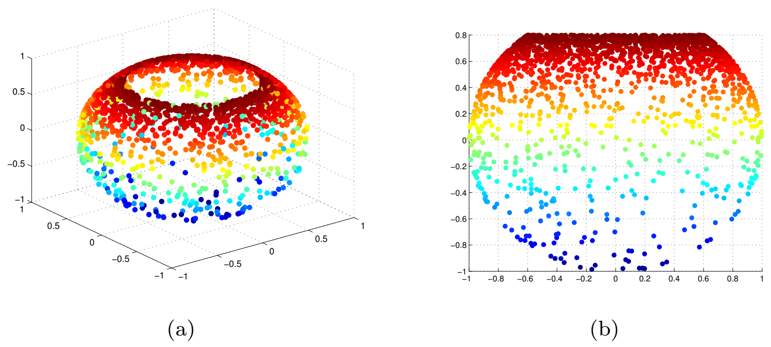


Figure 4.4: The 3-dimensional Fishbowl dataset. The data can be thought of as a sphere with the top chopped off and is named after the classic bowl shaped fish tank. (a) shows a 3-dimensional view of the dataset while (b) shows the 2-dimensional $y-z$ projection.

there is even less global information contained in the data as a result of the introduced discontinuity.

LTSA and PLML again out perform LLE and Isomap. At first sight both embeddings (Figure 4.3 (e-f)) are nearly identical with the difference that LTSA again normalises the embedding. However upon closer inspection there is a slight distortion in the centre of the PLML embedding along the lower connecting branch. The reason for this distortion becomes apparent when one considers that the models involved in the distortion represent two end points of the MST. Due to small accumulative errors being built up during the model merging procedure the end points of the MST do not exactly match so there is a slight distortion in the embedding.

4.2.3 Fishbowl

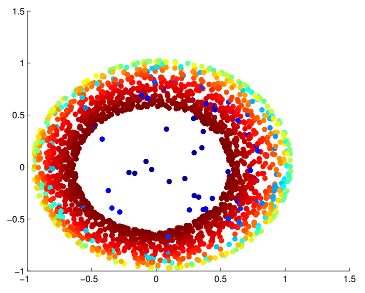
The Fishbowl dataset (Figure 4.4) consists of points sampled from a 3-dimensional sphere with the top section cut off so that it resembles a classic bowl shaped fish tank. The data is the stereographic projection in $\mathbb{R}^3$ from the plane $z = 0$

to the unit sphere (Silva and Tenenbaum, 2003b). A disk in the plane maps to a fishbowl structure under this map. It is worth noting from Figure 4.4 that the data is 'bunched' up non-uniformly near the rim of the bowl. Even though the points in the plane $z = 0$ are uniformly sampled, when projected onto the unit sphere this uniformity is lost. As pointed out in (Silva and Tenenbaum, 2003b) this non-uniform sampling can be difficult for spectral techniques such as Isomap to deal with.

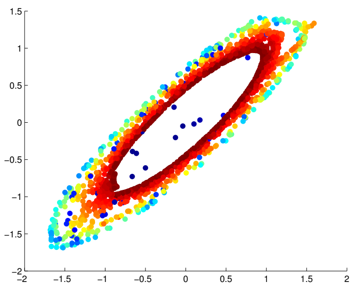
Another property of the dataset which makes it of interest is that it is a spherical manifold. This means that PLML will not be able to embed the manifold without introducing a cut. Due to the topological skeleton provided by the MST, circular, spherical, and any manifold with essential loops will be cut to produce an embedding. Although PLML introduces a cut to the manifold to produce an embedding, many manifold learning algorithms will be unable to successfully embed this dataset (Lee and Verleysen, 2005).

The embeddings produced by the different manifold learning algorithms on this Fishbowl dataset are shown in Figure 4.5. Isomap and LLE struggle to produce a good embedding of this dataset for two reasons. First, the non-uniform sampling makes it difficult for Isomap to properly recover the geodesic distances and also, as described above, the manifold contains essential loops.

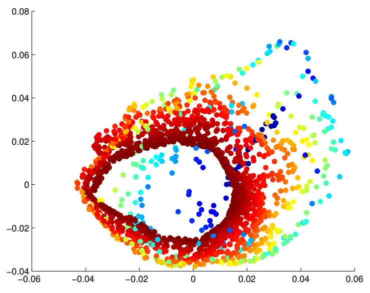
Both LTSA and PLML provide interesting embeddings of the data and it is difficult to differentiate which one is the correct embedding. LTSA recovers the original disk like structure that was cut out of the plane and projected onto the unit sphere. PLML on the other hand cuts the manifold and lays it out nicely in the low-dimensional space. Given that we know the way in which PLML operates (i.e. the manifold is cut if essential loops exist) then we can use the PLML embedding to infer that the high-dimensional points lie on a sphere as



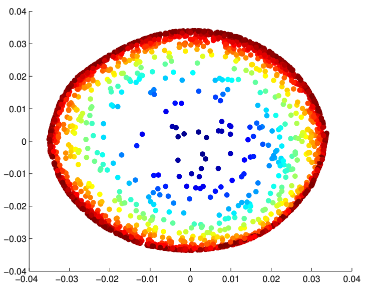
(a) PCA



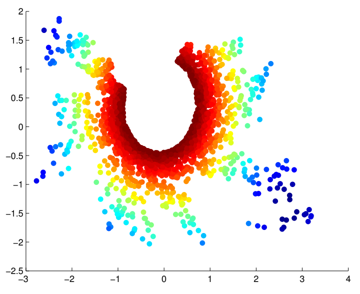
(b) Isomap



(c) LLE



(d) LTSA



(e) PLML

Figure 4.5: The 2-dimensional embeddings of the Fishbowl dataset.

we can imagine ‘re-stitching’ the embedding to form a sphere.

The Fishbowl manifold presents an interesting case as it is the first example so far where the two global alignment techniques - LTSA and PLML - differ drastically in results. With the Swiss Roll and broken Swiss Roll datasets both LTSA and PLML performed in much the same way. However the Fishbowl dataset, with its continuous curved structure and essential loops, shows how these two techniques differ. It is worth remembering that even though both LTSA and PLML are based on the idea of globally aligning local linear models their methodology is very different and this can often produce very different results.

4.2.4 Helix

The helix dataset is shown in Figure 4.6 and consists of a 1-dimensional curve embedded within 3-dimensional space where the tangent line to the curve at any point makes a fixed angle with the axis line. The helical structure is continuous, that is the two ends of the helix are joined together to make one continuous curve. This makes the true embedding of this dataset interesting as the intrinsic dimensionality of the data is 1, however it can be embedded in 2-dimensional space as a circle. This circular embedding is shown by the embeddings produced by Isomap, LLE and LTSA in Figure 4.7.

PCA provides a good approximation of the manifold by recovering the global and local curvature of the manifold. It is equivalent to an $x - y$ projection of the data (See Figure 4.6 (b)) and so some local distortions occur as a result of this linear projection. Isomap, LLE and LTSA manage to unravel the helix coil to a circle. Isomap and LLE perform particularly well at this while LTSA fails to uncover the structure but rather overlays different coils of the circle

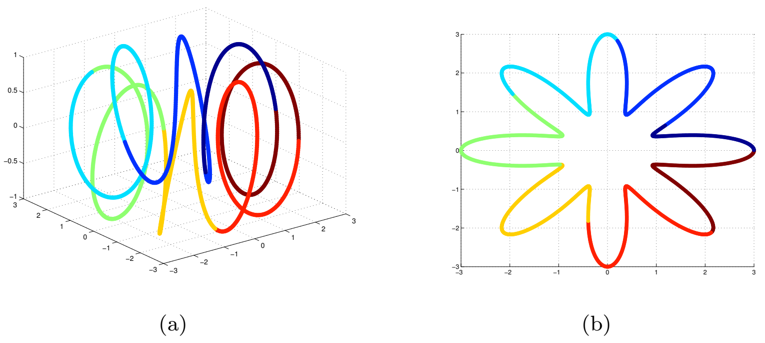
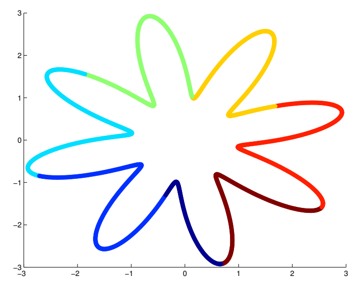


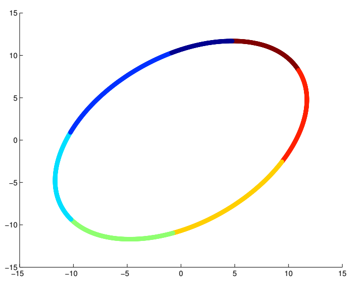
Figure 4.6: The 3-dimensional view, (a), and the $x - y$ projection, (b), of the Helix coil dataset.

on top of the others. As with the Fishbowl dataset, since the data contains essential loops PLML will not be able to successfully embed the helix data into 2-dimensions as a circle since a cut will be made in the manifold through use of the MST. This explains why the 2-dimensional embedding produced by PLML shown in Figure 4.7 is not circular. Rather it consists of each of the coils of the helix overlapping one another. The explanation of this is quite simple. The embedding produced by PLML is built in an iterative manner, that is one model is embedded into the global model and then the next model is embedded and so on. There is no 'all at once' global optimisation of the local models so the circle will never be recovered from the data as the helix will never be 'pulled out' into a circular shape. Rather PLML will simply follow the curvature of the manifold and embed the models according to this curvature.

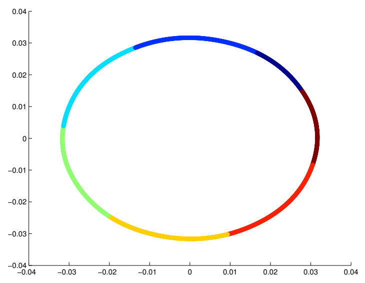
Since the Helix dataset can be thought of as a 1-dimensional curve there are two ways of embedding this dataset. The first, attempting to embed it as a circle in two-dimensions has already been discussed. We now move on to discuss the second way of embedding this dataset, by embedding it into a single dimension and attempting to recover the 1-dimensional structure of the data. The strength of PLML's cutting of the manifold becomes evident when we attempt to embed



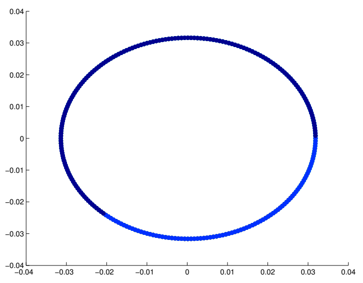
(a) PCA



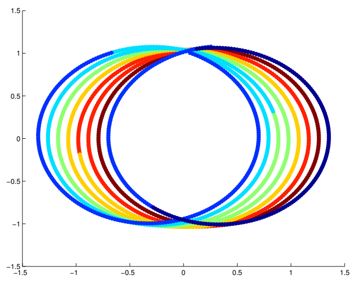
(b) Isomap



(c) LLE



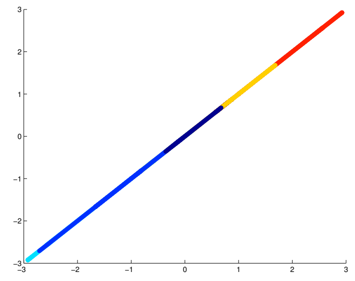
(d) LTSA



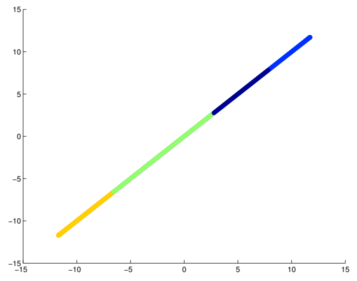
(e) PLML

Figure 4.7: The 2-dimensional embeddings of the Helix coil dataset.

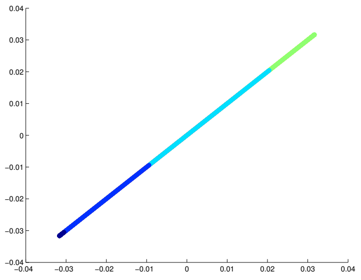
the helix in this way (Figure 4.8). All other algorithms that we are considering fail to properly embed the manifold into 1-dimension. They introduce distortions into the mapping by embedding sections of the manifold over the top of other sections. PLML on the other hand manages to successfully embed the manifold into 1-dimensional space without introducing any distortions.



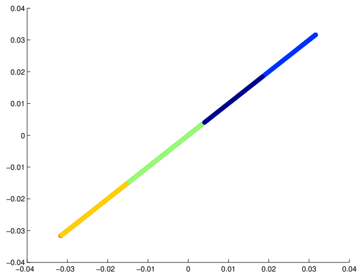
(a) PCA



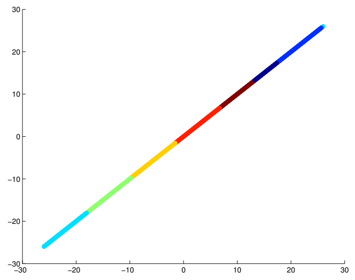
(b) Isomap



(c) LLE



(d) LTSA



(e) PLML

Figure 4.8: The 2-dimensional embeddings of the Fishbowl dataset.

4.3 PLML Detailed Analysis

In this section we provide a detailed analysis of the performance of the developed method, PLML, along with the 4 existing methods described previously (Isomap, LLE, LTSA and PCA). We focus solely on the Swiss Roll dataset and investigate the effects of sparsity, noise and parameter selection. The Swiss Roll exhibits many properties that make it ideal for our analysis. First, it is a non-linear dataset which means that the best results will be obtained by those methods that manage to learn its non-linear structure. Although we use PCA as a comparison algorithm, this non-linearity will mean that PCA becomes the base-linear method. It will never be able to 'un-roll' the data and so we can use PCA to represent the lower bounds of acceptability for an embedding.

The Swiss Roll also contains no essential loops. As previously shown, when encountering a dataset with essential loops (such as the Fishbowl or Helix) LTSA and PLML produce very different embeddings and others can even fail to embed the data (e.g. (Lee and Verleysen, 2005)). The Swiss Roll provides us with an environment in which the results of the methods are more stable, a fact which makes comparison far easier. Datasets with essential loops are re-visited in our detailed analysis of image data.

Finally, the Swiss Roll is the benchmark dataset for manifold learning. Although we only compare PLML against 4 other manifold learning algorithms it is easy to fit these results into the bigger picture of manifold learning by looking at how other methods perform on the Swiss Roll. If we were to use a less well known dataset for our detailed analysis we would not be able to do this. However, since many algorithms are presented with results based on their performance of learning the Swiss Roll dataset we can get a general idea of how well PLML will perform by comparing these results with those presented by other algorithms.

Throughout this analysis section our attention will be mainly fixed on the performance of the PLML algorithm. Although we will comment on the results produced by the other manifold learning algorithms our main interest is how well PLML performs and how it compares with the existing approaches.

4.3.1 Sparsity

The ability of manifold learning algorithms to deal with sparse data is one of the open problems in manifold learning (See Chapter 2). At a certain level of sparsity there may not be enough information contained within the data to reconstruct the manifold. However, algorithms should still be able to extract some meaningful structure from what little data there is. As the density of the data decreases and so the sparsity of the data increases, algorithms should be able to gracefully deteriorate with the sparseness of the data.

We use four different dataset sizes to represent the change of data from sparse to dense. The dataset sizes are $n = 250, 500, 1000$ and 2000 data points.

Figure 4.9 shows the Trustworthiness values for each dataset density value. For the sparse case ($n = 250$) PCA is the top performer with PLML and LLE close behind. Across all densities PCA is stable: producing similar results for each density value. This is apparent when we examine the standard deviation, Δ, between the Trustworthiness values for each density. For PCA, $\Delta_{\text{PCA}} = 0.0084$, where as the other techniques have standard deviation values of: $\Delta_{\text{ISO}} = 0.095$, $\Delta_{\text{LLE}} = 0.062$, $\Delta_{\text{LTSA}} = 0.164$ and $\Delta_{\text{PLML}} = 0.072$. The poor performance and high standard deviation of LTSA is due in part to neighbourhood size parameter selection. This is discussed in more detail later, but for now it is worth noting that when LTSA has the optimum parameter it performs well. This is exhibited when considering the dense data case for LTSA, a maximum

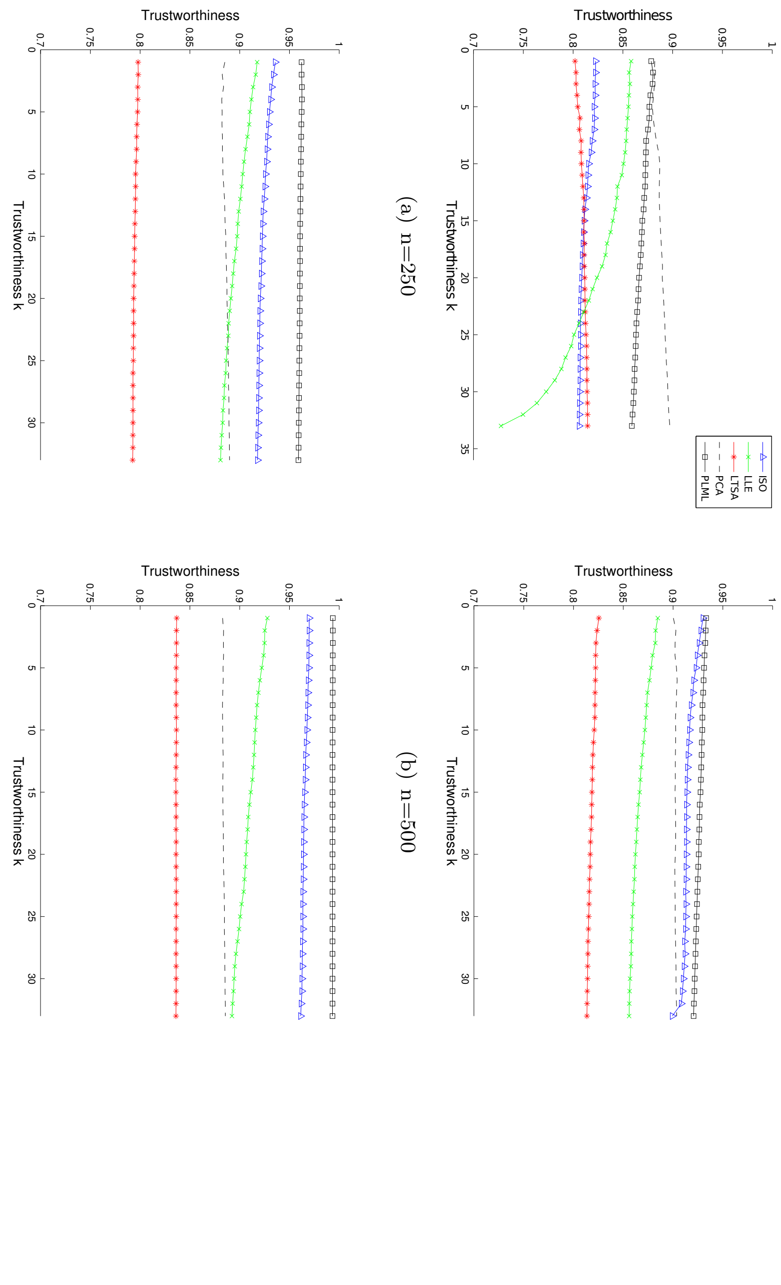


Figure 4.9: Trustworthiness values of the Swiss Roll dataset with average parameter values and $\sigma = 0$. The four figures show the results over the four different density values, $n = [250, 500, 1000, 2000]$.

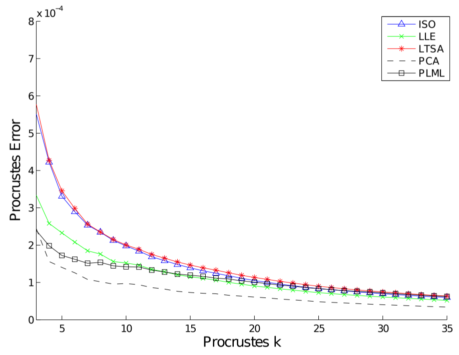
Trustworthiness value of $t = 0.991$ can be achieved with the correct parameter value ($k = 9$), however the Trustworthiness value significantly decreases when an incorrect parameter value is chosen (e.g. $t = 0.51$ when $k = 3$)[6].

At this point it is worth discussing the behaviour of Isomap and LLE at low densities. The Trustworthiness for LLE falls off as the size of the Trustworthiness region increases (Figure 4.9 (a)). This is due in part to the fact that LLE cannot recover the global properties of the data at such a low density.

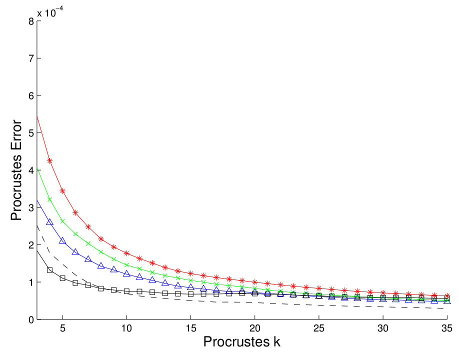
PLML performs well across all densities and values of Trustworthiness region with the exception of $n = 250$ and $n = 500$ where the Trustworthiness value tails slightly as the Trustworthiness region increases. The stability of Trustworthiness over a large range of k tells us something important about the performance of PLML. The models produced by PLML are discrete and contain no inter-model overlaps so it would be expected that up to the size of the local model (i.e. PLML's k value) the Trustworthiness measure would stay stable. However, as the Trustworthiness k value increases and exceeds the size of PLML's k value it is now more likely to see a drop in Trustworthiness as local relations are not guaranteed in the same way. The results in Figure 4.9 show that even across a neighbourhood size larger than PLML's k-value the Trustworthiness stays stable. This indicates that the local models are correctly aligned and positioned in the low-dimensional space as a result of PLML's merging process.

The Procrustes Error (PE) and Residual Mean Squared Error (RMSE) are shown in Figure 4.10 and Figure 4.11, respectively. Once again PCA performs well when compared to the other algorithms on the sparse data sets, especially when $n = 250$. The PE graphs all follow a similar pattern with the worst results occurring at low Procrustes k values and the results then levelling out as the

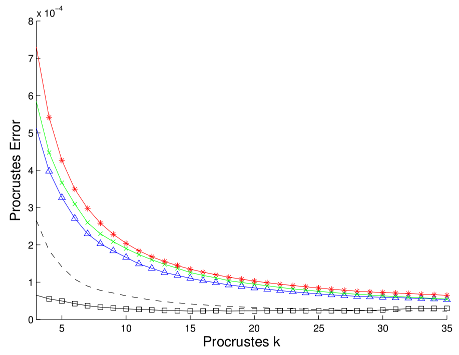
[6]These values are not represented in Figure 4.9 as the results are averaged over LTSA's neighbourhood size parameter and these good performances are outliers.



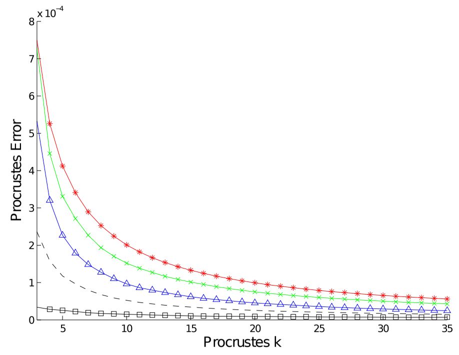
(a) n=250



(b) n=500

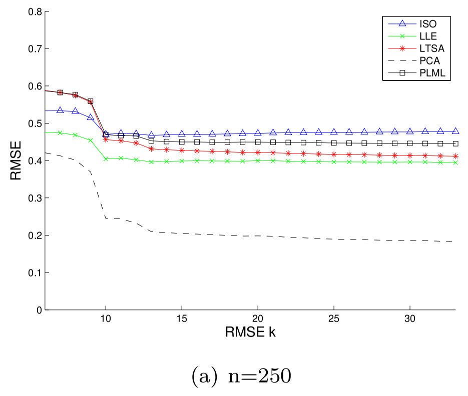


(c) n=1000

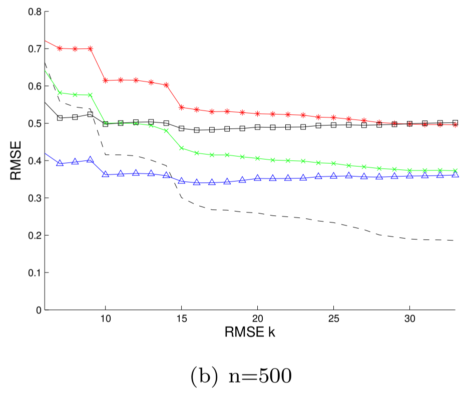


(d) n=2000

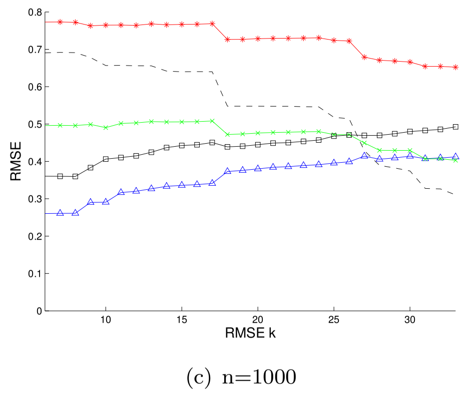
Figure 4.10: Procrustes Error values of the Swiss Roll dataset with average parameter values and $\sigma = 0$. The four figures show the results over the four different density values, $n = [250, 500, 1000, 2000]$.



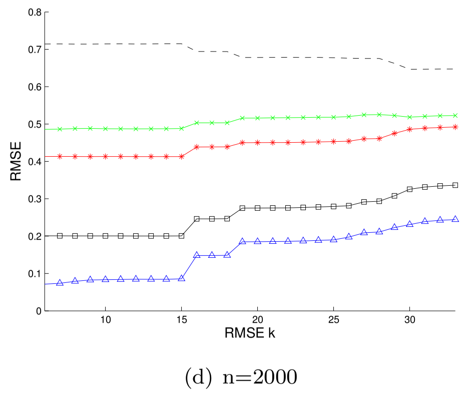
(a) n=250



(b) n=500



(c) n=1000

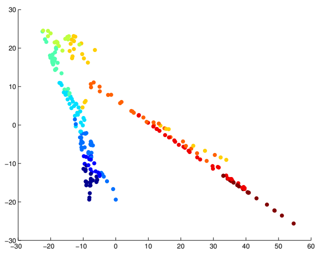


(d) n=2000

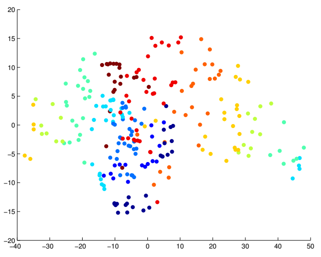
Figure 4.11: Residual Mean Squared Error values of the Swiss Roll dataset with average parameter values and $\sigma = 0$. The four figures show the results over the four different density values, $n = [250, 500, 1000, 2000]$.

Procrustes k value increases. As the data density increases PLML becomes more stable with the error value approaching 0 with the density set at $n = 2000$. This is not the case for the other algorithms. LTSA consistently performs the worst for the same parameter selection reasons described earlier. LLE and Isomap perform much the same at each density value. This indicates that distortions occur at a local scale as a result of performing manifold learning using either LTSA, LLE or Isomap with sub-optimal parameter values. PLML on the other hand is a lot more stable with different parameter values. This again shows that alignment of the local models is correctly performed as the inter-model distortion is low.

RMSE measures the retention of global information and as can be seen in Figure 4.11 all algorithms struggle to recover a good global embedding of the manifold when encountering sparse data. Considering that in this case PCA is the worst case as it will flatten the manifold and distort global distances then Figure 4.11 (a) shows that neither Isomap, LLE, LTSA or PLML can successfully recover the global structure of the manifold. This is shown in Figure 4.12 where the optimal embeddings of the $n = 250$ data for Isomap and PLML are shown.



(a) Isomap



(b) PLML

Figure 4.12: The embeddings produced by Isomap and PLML of the sparse Swiss Roll dataset with $n = 250$ and optimal parameter values of $k = 9$ and $k = 5$ chosen respectively.

Neither algorithm recovers the true manifold structure although Isomap does distort the local neighbourhoods (a fact reflected in the local error measures discussed previously). The performance of Isomap and PLML improves when the density is increased to $n = 500$ (Figure 4.11 (b)) however the performance of both tails off at the RMSE $k > 10$ mark. This trend of changing RMSE value at the RMSE $k > 10$ mark is due to the calculation of geodesics in the high-dimensional space. RMSE is based on the difference between geodesic distance as measured in the high-dimensional space and geodesic distance as measured in the low-dimensional space. The k value shown on the graphs in Figure 4.11 represents the size of the neighbourhood used to construct the geodesic distance graph. As this value increases more short circuits and incorrect distances will be introduced so the RMSE will become more unstable. This point is supported by the fact that the general shape of each curve in Figure 4.11 is very similar.

Isomap will always perform well when measuring the RMSE as it is the RMSE that the algorithm seeks to minimise. This aside, PLML still performs well especially when considering the densely sampled case of $n = 2000$. It is unsurprising that LLE does not perform as well as Isomap, LTSA or PLML as it is a local technique that does not guarantee to recover the global shape of the manifold.

Table 4.1 shows the average results of the different error measures over different densities with parameter values averaged and zero added noise. At low densities all non-linear algorithms are unstable and can often produce unpredictable results. This is shown by the fact that some of the results for $n = 500$ are worse than the results for $n = 250$ which is counter intuitive. One would expect the results to improve as the density of the data increases. However at lower density not only do the algorithms fail to recover the true manifold shape but the error measures used become harder to estimate.

		Trustworthiness	Procrustes Error $^{(\times 10^{-4})}$	RMSE
$n = 250$	ISO	0.792(±0.08)	1.525(±0.67)	0.482(±0.03)
	LLE	0.823(±0.07)	1.168(±0.40)	0.408(±0.05)
	LTSA	0.810(±0.00)	1.591(±0.14)	0.445(±0.03)
	PCA	0.889(±0.00)	0.750(±0.44)	0.229(±0.02)
	PLML	0.869(±0.00)	1.124(±0.25)	0.469(±0.01)
$n = 500$	ISO	0.915(±0.00)	0.982(±0.27)	0.362(±0.02)
	LLE	0.854(±0.09)	1.181(±0.55)	0.441(±0.05)
	LTSA	0.818(±0.00)	1.464(±1.18)	0.560(±0.04)
	PCA	0.903(±0.00)	0.627(±0.48)	0.314(±0.13)
	PLML	0.927(±0.00)	0.746(±0.42)	0.499(±0.04)
$n = 1000$	ISO	0.924(±0.00)	1.322(±0.50)	0.358(±0.04)
	LLE	0.896(±0.00)	1.488(±0.80)	0.473(±0.03)
	LTSA	0.795(±0.00)	1.688(±1.52)	0.729(±0.02)
	PCA	0.886(±0.00)	0.545(±0.53)	0.548(±0.12)
	PLML	0.961(±0.00)	0.291(±0.22)	0.440(±0.13)
$n = 2000$	ISO	0.966(±0.00)	0.865(±0.88)	0.156(±0.01)
	LLE	0.909(±0.00)	1.321(±0.91)	0.506(±0.01)
	LTSA	0.836(±0.00)	1.640(±1.42)	0.443(±0.03)
	PCA	0.884(±0.00)	0.453(±0.47)	0.688(±0.02)
	PLML	0.993(±0.00)	0.122(±0.11)	0.257(±0.06)

Table 4.1: Average results of Trustworthiness, Procrustes Error and RMSE over different density scales. The results were averaged over parameter value, error metric parameter value and taken with zero added noise. The standard deviations as a result of this averaging are shown.

The general trend in results is that as the density increases the algorithms become more stable as they are able to better recover the underlying manifold. PCA performs much the same regardless of density which lends weight to the argument that PCA can be used as a good initial approximation when using manifold learning for exploratory data analysis (Lee and Verleysen, 2007).

The results on different densities show that all non-linear algorithms struggle when the data density is low ($n < 1000$). This is unsurprising as there is often not enough local and global information contained in the sparse data to approximate the shape of the manifold correctly at a local and global scale. PCA on the other hand performs consistently regardless of density and outperforms the non-linear techniques at low densities. However, when the density increases ($n \geq 1000$) the non-linear algorithms begin to outperform PCA. With $n \geq 1000$ PLML becomes the top performer as it is able to recover both the local and global properties of the manifold. Even though Isomap just out performs PLML when it comes to RMSE at high densities, PLML manages to out perform Isomap when observing local stability.

As well as this, PLML degrades gracefully as the density decreases and sparsity increases. By observing the change of values in Table 4.1 we can see that apart from one slight outlier with RMSE at $n = 500$ the performance of PLML improves as the density increases. Similarly we know that as the density decreases then the performance of PLML will also decrease, but not to the extent to make it unusable.

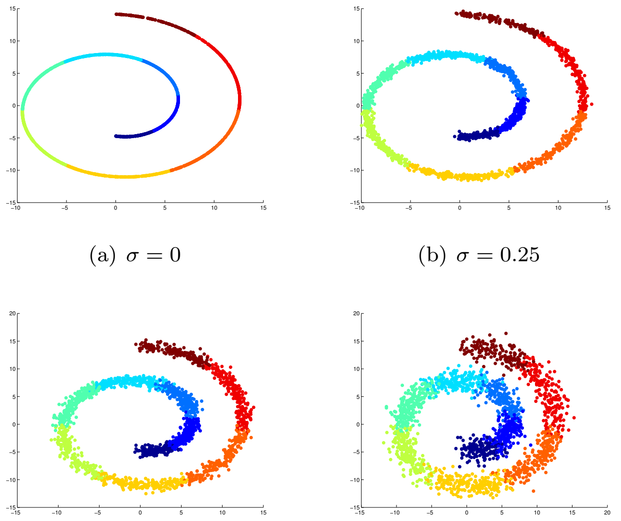
With high-density data PLML is able to perform well at both a local and global scale. It is able to outperform all the other manifold learning algorithms we are comparing. As the density of the data decreases and we encounter increased sparsity then the performance of PLML begins to drop. However, the performance of PLML at low-density is still comparable at a local scale to that of PCA meaning that it meets the lower bounds of performance. Globally, at low density, PLML is unable to recover the true manifold structure. This is not however specific only to PLML. All other manifold learning algorithms fail to uncover the global structure at low-density. As such, as long as the data is well sampled then PLML will outperform the other manifold learning algorithms.

4.3.2 Noise

Many real world datasets are inherently noisy and so it is important for algorithms that handle such data to cope with noise in a well defined manner. Within the field of manifold learning, noise can often lead to distorted or even completely unusable embeddings. As such the ability of manifold learning algorithms to deal with noise is an important topic. In this section we investigate the effect of noise on the performance of the 5 manifold learning algorithms we are comparing.

The noise model used throughout these experiments is that of Gaussian noise with a mean of 0 and standard deviation in the range of $\sigma = [0, 1]$. Figure 4.13 illustrates the effect that noise has on the Swiss Roll with $n = 2000$. Although with $\sigma = 1$ the Swiss Roll does not completely degrade it is still difficult for many manifold learning algorithms to pick up the true structure of the manifold with this amount of noise. Throughout the discussion on noise we use $n = 2000$ unless otherwise stated. This is so as to factor out the effect of density and concentrate solely on the effect of noise.

Figure 4.14 shows the effect of noise on algorithm performance on the Swiss Roll dataset. The values for each error measure were obtained by averaging both the error measure and manifold learning parameter values (See Table 4.2 for



(a) $\sigma = 0$

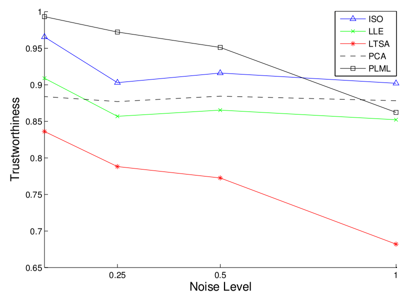
(b) $\sigma = 0.25$

(c) $\sigma = 0.5$

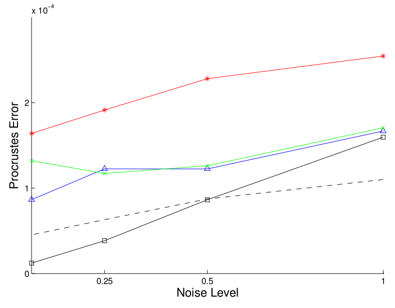
(d) $\sigma = 1$

Figure 4.13: The different amounts of noise used in the experiments. Gaussian noise is added at each noise level with different standard deviations in the range $\sigma = [0, 1]$

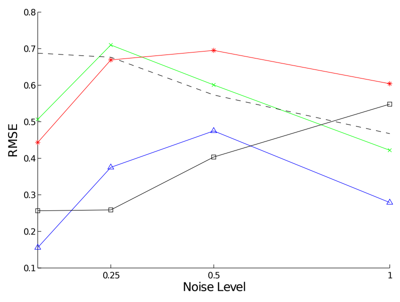
standard deviations). For each error measure PLML degrades with noise; as the noise increases the performance of PLML decreases. With the exception perhaps of PCA, PLML is the only algorithm that shows a well defined and predictable decrease in performance. The other algorithms are more unpredictable in the presence of noise. For example, consider the RMSE value for LLE (Figure 4.14 (c)). At $\sigma = 0$, LLE records an average equal to $\text{RMSE}_{\text{LLE}} = 0.506$, at $\sigma = 0.25$ this value increases to $\text{RMSE}_{\text{LLE}} = 0.71$ indicating a decrease in the global quality of the embedding. However as the noise increases the value of σ drops again indicating that LLE produces a better global approximation. This tells us two things. First, LLE is unable to recover a good global approximation of the manifold even at a low noise level (the value of RMSE_{LLE} at $\sigma = 0$ is higher than the value of RMSE_{ISO} at $\sigma = 1$). Secondly, it tells us that LLE is unstable in the presence of noise. Even though it is unable to gain a



(a) Trustworthiness



(b) Procrustes Error



(c) RMSE

Figure 4.14: The effect of noise on a densely sampled Swiss Roll ($n = 2000$) when measuring average Trustworthiness, Procrustes Error and RMSE. The values were obtained at each noise level over an average of error measure parameters and manifold learning neighbourhood size parameters.

good global approximation of the manifold it does not improve as the amount of noise is reduced. The global stability of PLML on the other hand decreases in a deterministic fashion as the noise increases. That is, as you move up from one noise level to another the quality of the embedding decreases.

So although the overall performance of PLML does decrease as the noise level increases, it does so in a well defined manner. From Figure 4.14 it is clear that at a local scale most algorithms perform in a defined manner. From the Trustworthiness and Procrustes Error values it is apparent that as the noise increases the performance of an algorithm either drops (as is the case for all algorithms when measuring the Procrustes Error) or stabilises (as is the case for Isomap, LLE and PCA when measuring the Trustworthiness). Although PLML does not perform as well as some of the other algorithms at high noise values it is still useful to know that the performance decreases in a well defined manner and does not suddenly drop at a certain noise level.

We now move on to discuss why the performance of PLML drops in the presence of noise. We suggest that there is one main reasons why PLML's performance

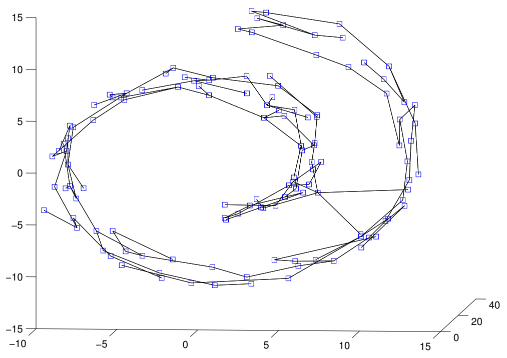


Figure 4.15: The effect of noise on the creation of the MST within the PLML algorithm.

drops in the presence of noise. The noise affects the clustering step and so in turn leads to not only incorrect models being formed but also an incorrect MST being produced. If the clustering step fails to produce accurate local models of the data then it is unlikely for the PLML algorithm to recover a true embedding of the manifold. Figure 4.15 shows an MST created using the PLML algorithm with $n = 2000$ and $\sigma = 1$. It is clear to see that short-circuits have been introduced into the MST that cut across the manifold. These short-circuits will introduce distortions and overlaps into the embedding as the merging step will merge two models that are not topological neighbours. This is why the Trustworthiness decreases: overlaps cause points to jump between non-connected neighbours. Similarly the RMSE increases because short-circuits distort the true distances between points and models. The Procrustes Error increases but not to such an extend as is shown in Trustworthiness and RMSE. This is due in part to the models retaining their local geometric properties even if the MST is not correctly formed.

Since PLML performs well at mid to low noise (e.g. $\sigma < 0.5$) it could be worthwhile applying a noise reduction algorithm prior to using PLML so as to attempt to factor out any of the effects that noise may have on the performance.

4.3.3 Parameter Selection

Although methods exist to estimate the optimal parameters for many manifold learning algorithms (See Chapter 2) it is still worthwhile to note how algorithms perform with incorrect parameter choices. In this study there are two parameters to be investigated. The first, neighbourhood size, is applicable to all the non-linear dimensionality reduction methods we are comparing. The second, the starting model, is applicable only to PLML. Since PCA is a parameterless method we do not include it in the following discussions.

		Trustworthiness	Procrustes Error $(\times 10^{-4})$	RMSE
$\sigma = 0$	ISO	0.966($\pm$0.00)	0.865($\pm$0.88)	0.156($\pm$0.01)
	LLE	0.909($\pm$0.00)	1.321($\pm$0.91)	0.506($\pm$0.01)
	LTSA	0.836($\pm$0.00)	1.640($\pm$1.42)	0.443($\pm$0.03)
	PCA	0.884($\pm$0.00)	0.453($\pm$0.47)	0.688($\pm$0.02)
	PLML	0.993($\pm$0.00)	0.122($\pm$0.11)	0.257($\pm$0.06)
$\sigma = 0.25$	ISO	0.903($\pm$0.01)	1.226($\pm$0.47)	0.375($\pm$0.03)
	LLE	0.857($\pm$0.00)	1.171($\pm$0.51)	0.710($\pm$0.01)
	LTSA	0.788($\pm$0.01)	1.914($\pm$1.30)	0.669($\pm$0.02)
	PCA	0.877($\pm$0.00)	0.631($\pm$0.76)	0.677($\pm$0.03)
	PLML	0.972($\pm$0.00)	0.384($\pm$0.14)	0.259($\pm$0.05)
$\sigma = 0.5$	ISO	0.916($\pm$0.00)	1.224($\pm$0.37)	0.475($\pm$0.08)
	LLE	0.865($\pm$0.00)	1.262($\pm$0.66)	0.601($\pm$0.02)
	LTSA	0.773($\pm$0.00)	2.281($\pm$1.28)	0.695($\pm$0.02)
	PCA	0.884($\pm$0.00)	0.872($\pm$0.91)	0.573($\pm$0.12)
	PLML	0.951($\pm$0.00)	0.863($\pm$0.89)	0.404($\pm$0.05)
$\sigma = 1$	ISO	0.902($\pm$0.00)	1.668($\pm$0.34)	0.279($\pm$0.03)
	LLE	0.852($\pm$0.00)	1.706($\pm$0.67)	0.442($\pm$0.03)
	LTSA	0.682($\pm$0.00)	2.546($\pm$1.18)	0.604($\pm$0.14)
	PCA	0.878($\pm$0.00)	1.101($\pm$1.00)	0.194($\pm$0.01)
	PLML	0.862($\pm$0.00)	1.596($\pm$0.00)	0.548($\pm$0.04)

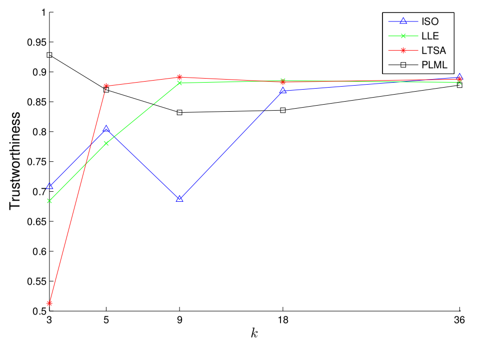
Table 4.2: Average results of Trustworthiness, Procrustes Error and RMSE over different noise levels. The results were averaged over parameter value, error metric parameter value and taken with $n = 2000$. The standard deviations as a result of this averaging are shown.

Neighbourhood Size

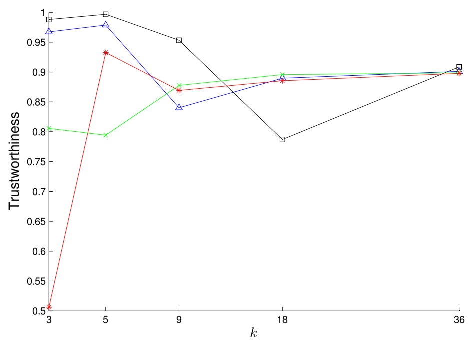
The neighbourhood size parameter, k, has various uses depending on which manifold learning technique is being used. For Isomap the neighbourhood size parameter determines the number of nearest neighbours to connect when forming the neighbourhood graph. For LLE the neighbourhood size parameter determines the number of points to include in the reconstruction of a sample from its neighbours. For LTSA and PLML the neighbourhood size parameter determines the size of the local tangent spaces or models. With this in mind it is expected that with a small k value, Isomap will fail to recover a faithful embedding as it will produce a disconnected graph, LLE will not recover enough local information to reconstruct the data-points properly, whereas the local models produced by LTSA and PLML will not contain enough local information to gain a proper approximation of the structure of the manifold. Similarly, if the value of k is too large then Isomap's neighbourhood graph will short-circuit leading to incorrect geodesics. LLE, LTSA and PLML will over approximate the linearity of the local models and so produce incorrect mappings because they attempt to enforce linear constraints on parts of the manifold which are in fact non-linear.

The investigation into the effect of neighbourhood size parameter is broadly split into two categories. First, we deal with how neighbourhood size and data density are related and then secondly we deal with how neighbourhood size and noise are related. Each of these are dealt with separately below.

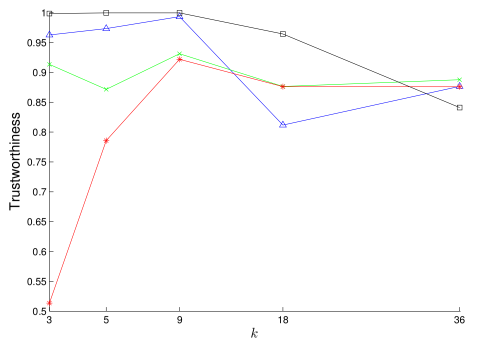
Figure 4.16 shows the results of changing the neighbourhood size parameter over a range of different densities. The first point to note is that the effect of changing the neighbourhood size parameter decreases as the sample density increases. This is especially true for Isomap and PLML where the standard deviations for $n = 250$ are $\Delta_{\text{ISO}} = 0.0921$ and $\Delta_{\text{PLML}} = 0.0389$ respectively.



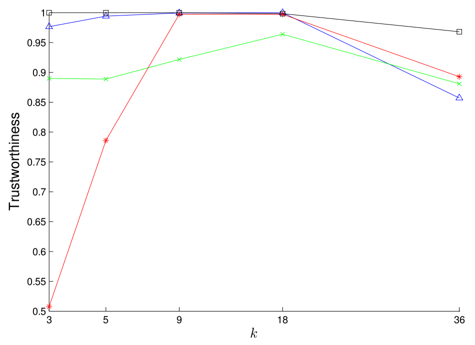
(a) n=250



(b) n=500

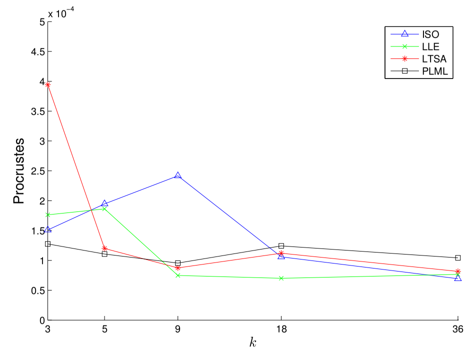


(c) n=1000

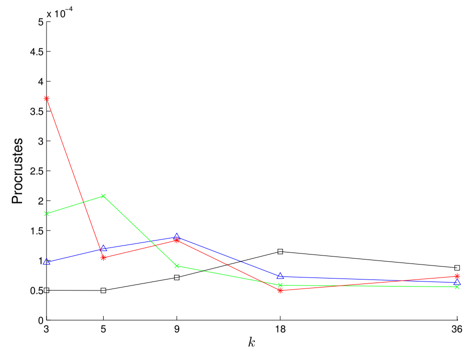


(d) n=2000

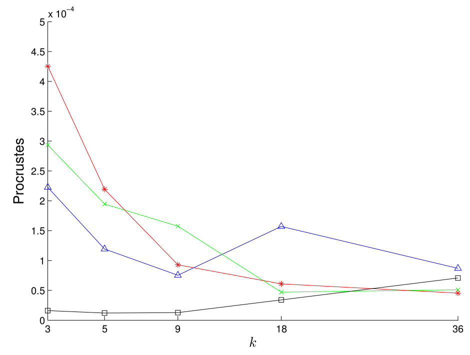
Figure 4.16: The change in Trustworthiness over a range of densities when changing the k neighbourhood size parameter. These results were obtained over a range of Trustworthiness size parameter and with $\sigma = 0$.



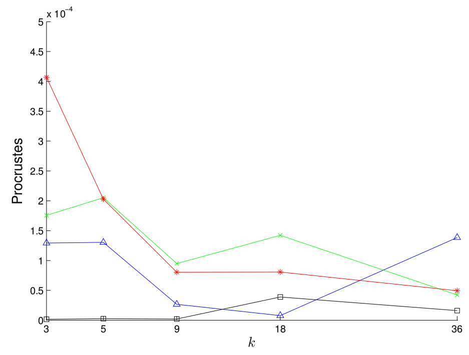
(a) n=250



(b) n=500

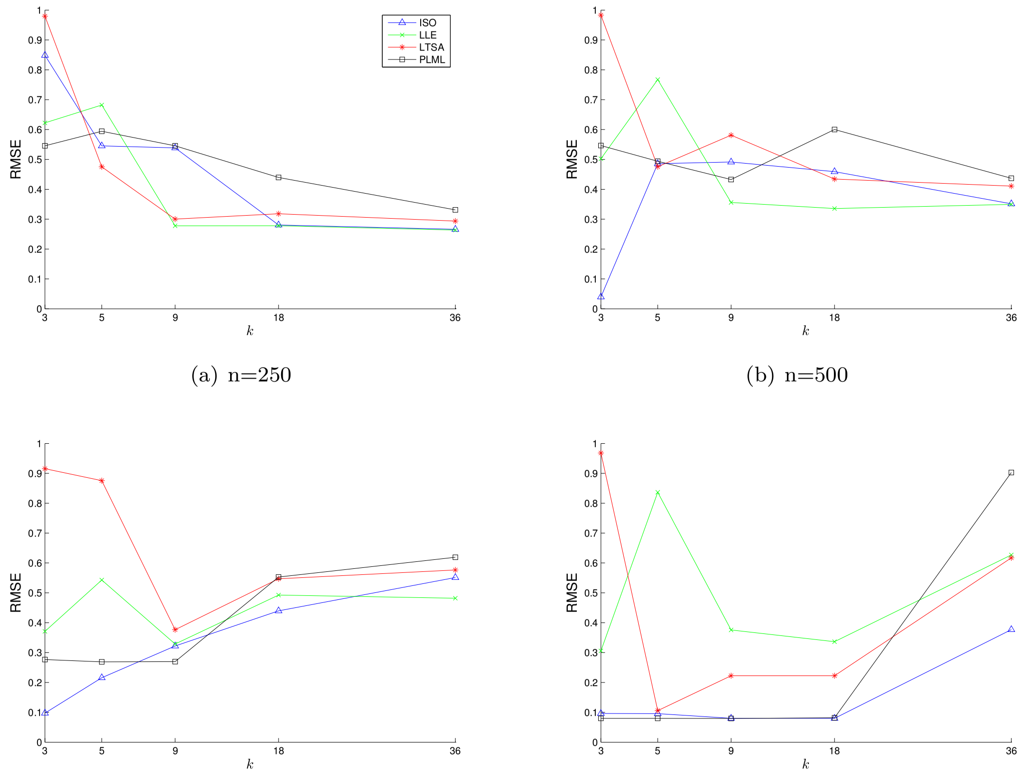


(c) n=1000



(d) n=2000

Figure 4.17: The change in Procrustes Error over a range of densities when changing the k neighbourhood size parameter. These results were obtained over a range of Procrustes Error neighbourhood size parameter and with $\sigma = 0$.



(a) n=250 (b) n=500

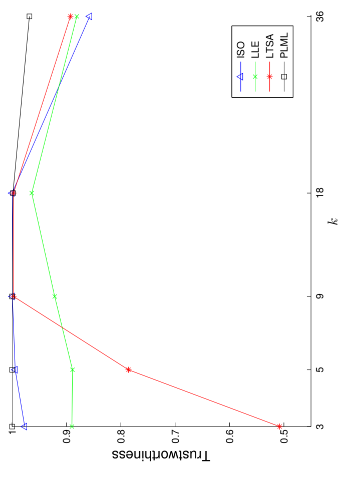
(c) n=1000 (d) n=2000

Figure 4.18: The change in RMSE over a range of densities when changing the k neighbourhood size parameter. These results were obtained over a range of RMSE neighbourhood size parameter and with $\sigma = 0$.

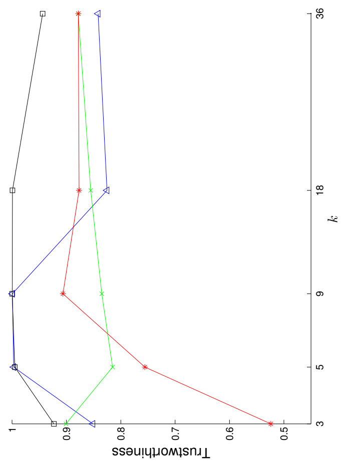
These values drop when $n = 2000$ to $\Delta_{\text{ISO}} = 0.0613$ and $\Delta_{\text{PLML}} = 0.0141$. When compared to LTSA's standard deviations of $\Delta_{\text{LTSA}} = 0.163$ for $n = 250$ and $\Delta_{\text{LTSA}} = 0.203$ it is clear that Isomap, PLML, and to a certain extent LLE, follow the pattern of stabilising the embedding quality over different parameters as the number of samples increases. Just as increased density stabilised the results obtained over average parameter values, so increased density stabilises the parameter values. Figure 4.17 and Figure 4.18 show the Procrustes Error and RMSE results obtained over different k values for different densities. What is interesting to note is that although the results do still partially stabilise as the density increases there exists an optimal range of parameters in a way that is not quite so obvious in the Trustworthiness results.

For nearly all techniques, bar perhaps LTSA, this optimal parameter value occurs in the range $4 \leq k \leq 14$. This could be due in part to the fact that within this range the local linearity assumption holds. At values smaller than this there is not enough local information to properly construct the linear basis of the neighbourhood. Similarly, at k values larger than this the linearity assumption fails to hold because the neighbourhood is over estimated and so lies across a non-linear patch of the manifold.

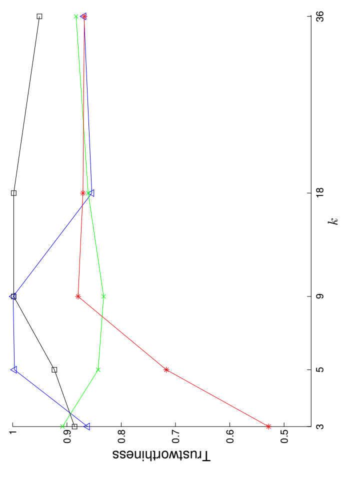
The effect of noise on the choice of parameter value is shown in Figure 4.19, Figure 4.20 and Figure 4.21. Considering the case of PLML only, these results show that as the noise increases so does the optimal value of k. For example, the optimal k value for Trustworthiness at $\sigma = 0$ occurs in the range $3 \leq k \leq 9$. Where as this optimal value increases at $\sigma = 1$ to the range $18 \leq k \leq 36$. The story is the same for Procrustes Error and RMSE (Figure 4.20 and Figure 4.21). This leads us to draw a general conclusion: as the noise increases we should increase the neighbourhood size parameter to attempt to counter the effect of noise. At a certain level this increase in k will have a negative rather



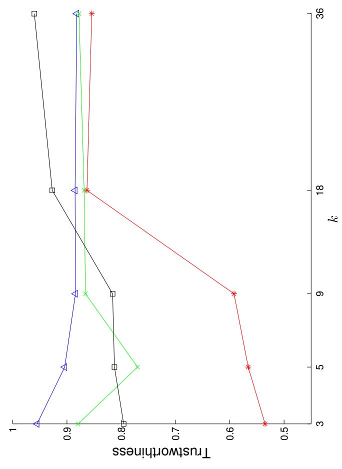
(a) $\sigma = 0$



(b) $\sigma = 0.25$

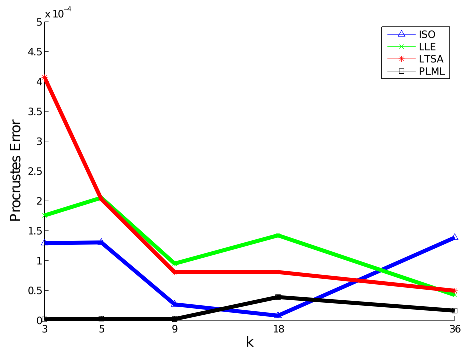


(c) $\sigma = 0.5$

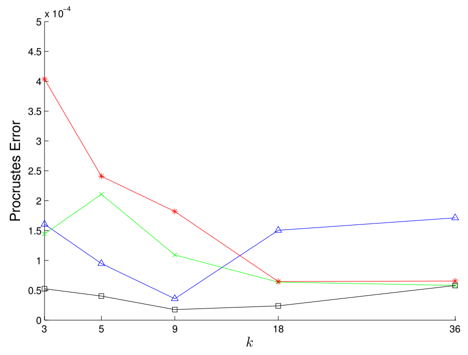


(d) $\sigma = 1$

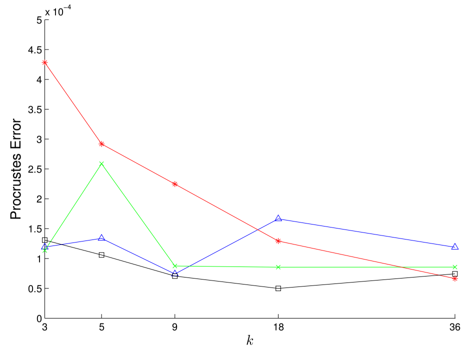
Figure 4.19: The change in Trustworthiness over a range of noise values when changing the k neighbourhood size parameter. These results were obtained over a range of Trustworthiness size parameter and with $n = 2000$.



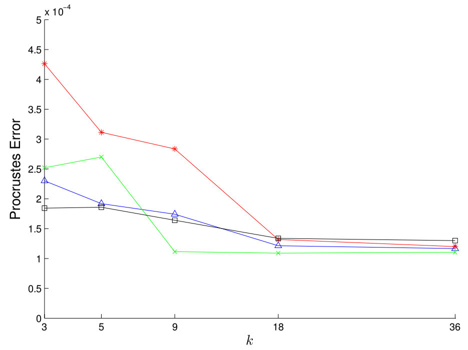
(a) $\sigma = 0$



(b) $\sigma = 0.25$

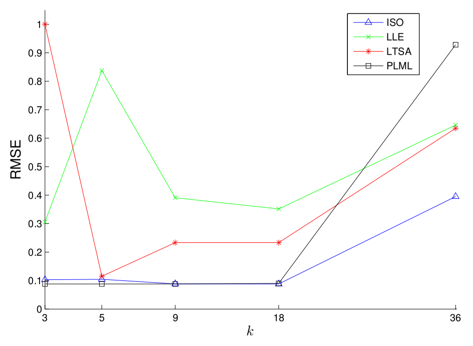


(c) $\sigma = 0.5$

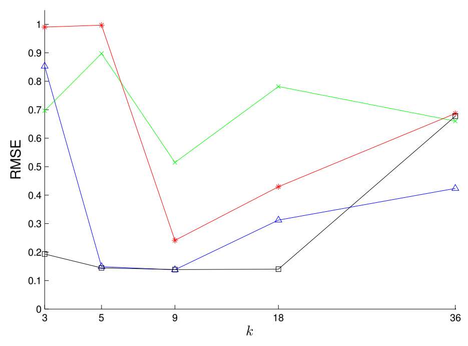


(d) $\sigma = 1$

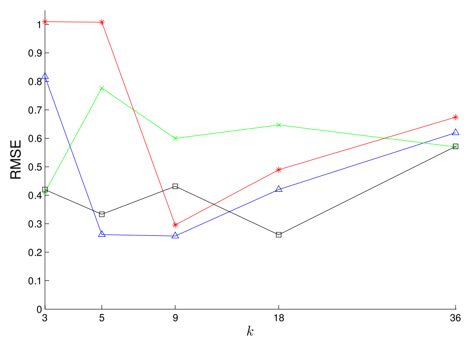
Figure 4.20: The change in Procrustes Error over a range of noise values when changing the k neighbourhood size parameter. These results were obtained over a range of Trustworthiness size parameter and with $n = 2000$.



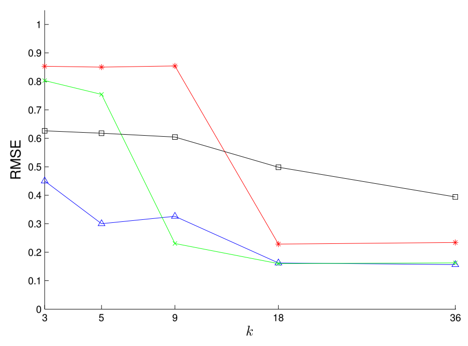
(a) $\sigma = 0$



(b) $\sigma = 0.25$



(c) $\sigma = 0.5$



(d) $\sigma = 1$

Figure 4.21: The change in RMSE over a range of noise values when changing the k neighbourhood size parameter. These results were obtained over a range of Trustworthiness size parameter and with $n = 2000$.

than positive effect (as can be seen at the higher k values for the higher noise value figures).

However, we have seen when discussing the effect of density on the choice of k that large values of k with sparse data can often have detrimental effects to the quality of the embedding. Therefore a careful trade off needs to take place between correct parameter selection based on the density and noise of the dataset. It is difficult to draw a general rule as it is likely to be data dependent but it is worthwhile to note that it will be difficult to select an optimal parameter for a low density, high noise data set.

Considering the two different problems of varying k with density and varying k with noise we can see that it is difficult to formulate a general rule as to the correct choice of parameter. However, we are able to develop some general principles which are specific to the PLML algorithm. If the data density is high then the effect that the neighbourhood size parameter plays in the performance of the PLML algorithm decreases. At lower densities a smaller parameter value should be used to achieve better results. This is due in part to the fact that the number of samples in a neighbourhood relative to the overall number of samples decreases so the parameter value should also decrease. However in the presence of high noise this neighbourhood size value should increase to compensate for the effect of noise. As such there is a trade-off involved in the selection of the correct neighbourhood size parameter. The stability and predictable performance of PLML at different densities and noise levels should make this parameter search easier.

Starting Model

The PLML specific parameter which we now investigate is the starting model used in the MST traversal. The assumption we wish to prove is that a random model can be chosen to initiate the traversal and choosing another starting model will not change the embedding result. To address this we choose 15 different starting points from the Swiss Roll each at a different position either around the edge or along the centre of the manifold (Figure 4.22). We then measure the Trustworthiness, Procrustes Error and RMSE for each of these embeddings. To cancel out the effect of noise, density and neighbourhood size parameter we use a Swiss Roll with $n = 2000$, $\sigma = 0$ and set PLML's k value to $k = 9$ (this value falls within the optimal range of parameter values discussed above for a high-density, low noise dataset).

The starting points are labelled as shown in Figure 4.22 according to their position relative to the low-dimensional embedding. So the top left point is called TL, the middle point is called MC (for Middle Centre) and the Bottom

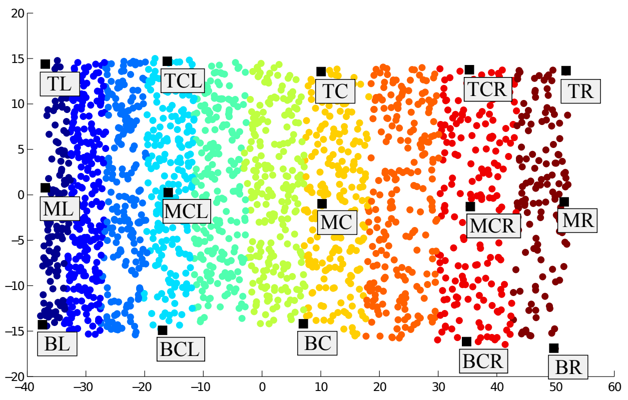


Figure 4.22: The different starting points used to test whether the starting model affects the low-dimensional embedding. The points are shown as their respective positions on the low-dimensional embedding.

	Trustworthiness	Procrustes Error $^{(\times 10^{-6})}$	RMSE
TL	0.999($\pm$0.00)	2.967($\pm$0.96)	0.100($\pm$0.07)
ML	0.999($\pm$0.00)	2.437($\pm$0.67)	0.099($\pm$0.07)
BL	0.999($\pm$0.00)	2.855($\pm$0.84)	0.100($\pm$0.07)
TCL	1.000($\pm$0.00)	2.889($\pm$0.91)	0.100($\pm$0.07)
MCL	0.999($\pm$0.00)	2.525($\pm$0.83)	0.100($\pm$0.07)
BCL	0.999($\pm$0.00)	8.198($\pm$3.35)	0.101($\pm$0.07)
TC	0.999($\pm$0.00)	2.496($\pm$0.63)	0.100($\pm$0.07)
MC	1.000($\pm$0.00)	2.037($\pm$0.52)	0.100($\pm$0.07)
BC	0.999($\pm$0.00)	4.667($\pm$5.40)	0.100($\pm$0.07)
TCR	0.999($\pm$0.00)	3.349($\pm$0.74)	0.101($\pm$0.07)
MCR	0.999($\pm$0.00)	3.717($\pm$1.37)	0.100($\pm$0.07)
BCR	0.999($\pm$0.00)	2.274($\pm$0.39)	0.100($\pm$0.07)
TR	0.999($\pm$0.00)	2.823($\pm$0.83)	0.101($\pm$0.07)
MR	1.000($\pm$0.00)	2.351($\pm$0.50)	0.100($\pm$0.07)
BR	0.999($\pm$0.00)	3.205($\pm$0.97)	0.100($\pm$0.07)
Δ	$\pm$0.00	$\pm 2.31 \times 10^{-6}$	$\pm$0.07

Table 4.3: Average results of Trustworthiness, Procrustes Error and RMSE over different starting positions. The values were obtained on a Swiss Roll with $n = 2000$ and $\sigma = 0$ with PLML $k = 9$.

Right is called BR.

Table 4.3 shows the results of testing the embedding quality when starting from the 15 different starting models as shown in Figure 4.22. The bottom row of Table 4.3 shows the standard deviation across all starting models. A low value indicates that the starting model has no effect on this quality measure. With this in mind we can see that the starting model has no effect on Trustworthiness as the average standard deviation is 0.00. This is not the case for the Procrustes Error and RMSE. However these standard deviations can be explained. First, the standard deviation for RMSE simply reflects the standard deviation within the measurement of RMSE for each starting model. The value of 0.07 is constant across all models and so does not reflect any change between the RMSE when starting at different models but rather the standard deviation from within each embedding. For Procrustes Error the story is much the same however there are some slight outliers, most notably BCL, BC, TCR and MCR. However for each of these starting positions the standard deviation is high within the embedding measurement suggesting that there are outlier measurements rather than a drastic change of Procrustes Error for that specific starting model.

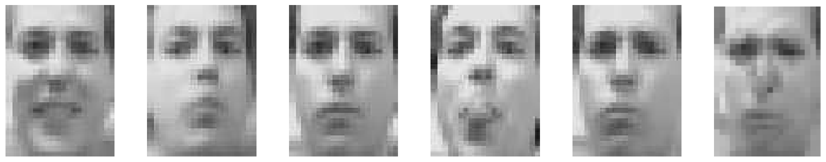
We have shown that although there is slight deviation in the results when choosing different starting models these deviations do not create drastically different embeddings. As such the starting model does not have enough effect on the resulting embedding to warrant an ideal starting point. Rather we can select a starting model at random and be sure that the embedding quality will remain the same regardless of which random model we choose.

4.4 Image Data

Learning the low-dimensional manifold of image data presents an interesting and 'real world' problem for manifold learning algorithms. Testing manifold learning algorithms on image data has become one of the corner stones of this research field (e.g. Tenenbaum et al. (2000); Roweis and Saul (2000); Verbeek (2006); Weinberger and Saul (2006a)). Typically the image data will be synthetically produced so as to limit the number of intrinsic dimensions of the data. For example, the ISO Faces dataset (Tenenbaum et al., 2000) consists of a 3-dimensional head model changing in left-right and top-down pose with the lighting conditions also changing. This means that the parameterisation of the intrinsic manifold is constrained to the head pose and lighting direction. More complex data can be found in the form of the Frey faces dataset (Roweis and Saul, 2000). This dataset is a set of frames taken from a video sequence of a face undergoing different facial expressions. These expressions are the parameters of the low-dimensional manifold.

In all of the image data examples the raw pixel information is used as the input features. Each image represents a single point in high-dimensional space. So if we have a set of 200 images of size 32×32 pixels then we will have a set of 200 points in $\mathbb{R}^{1024}$. This means that we are attempting to learn the true intrinsic manifold of the data rather than the manifold created as a result of extracting a set of image features.

The analysis of this image data is difficult as there is often no ground truth that describes what the embedding should look like or what properties the embedding should have. Rather the analysis of the image dataset is heavily reliant on visual assessment of the embedding. For this reason we perform a visual, qualitative, assessment of the image datasets rather than a quantitative



(a) Happy (b) Pouting (c) Neutral (d) Tongue (e) Sad (f) Angry

Figure 4.23: The 6 expressions that parameterise the Frey faces dataset. These expressions were extracted from the PLML embedding as the mean image of each manually segmented expression region

analysis. For each of the image datasets used we follow the same methodology. First, we examine the topology of the data according to the model based MST. The MST is laid out using the Graph Viz package[7] and the mean image of each model is displayed. This allows us to investigate the connectedness of the dataset separate from the embedding. As well as investigating the MST we also investigate the 2-dimensional embedding produced by the PLML algorithm and discuss the results obtained. The pros and cons of the performance of the PLML algorithm on each dataset is also discussed.

We use 3 different image based datasets: the Frey faces dataset (Roweis and Saul, 2000), the ISOFaces data (Tenenbaum et al., 2000) and the COIL-20 data (Nene et al., 1996). Each of these datasets contain an interesting and potentially difficult manifold structure. For each dataset we discuss the expected and learnt manifold structure as well as the estimated intrinsic dimensionality of each dataset.

4.4.1 Frey Faces

The Frey faces dataset (Roweis and Saul, 2000) consists of a sequence of images showing a face undergoing different facial expressions (Figure 4.23). There are

[7]The open source GraphViz library is available from: `http://www.graphviz.org` (*Link checked 22/07/11*)

1965 images each of size 20×28 pixels, meaning the data lies in $\mathbb{R}^{560}$. The only change in the data between frames is the facial expression with some left-right and up-down pose variation. The lighting is fixed. However, the amount of time spent on each facial expression varies. So for example, the neutral facial expression is over represented, whereas the tongue expression is under represented. This raises questions as to what is the true intrinsic structure and dimensionality of the data. In the literature this dataset is often reduced to 2-dimensions and then paths are traced through the embedding to show how the variation of expressions can be interpolated (e.g. Roweis and Saul (2000); Roweis et al. (2002); Goldberg and Ritov (2009)). This embedding to 2-dimensions however makes many assumptions about the intrinsic structure of the data that may not necessarily be true. First, the intrinsic dimensionality of the data is not 2-dimensions. An average of intrinsic dimensionality estimators reaches the intrinsic dimension value of 7-dimensions[8]. This means that reducing to 2-dimensions will most likely introduce discontinuities and distortions into the embedding. Secondly, the idea of tracing a path through the embedding to show expression interpolation does not equate to a successful embedding. In the extreme case a carefully selected path could be mapped through a random projection of the data and structure could be inferred.

The question of what constitutes a successful embedding of the Frey faces data remains an open question. We wish to suggest that if reducing the data to 2-dimensions, as we do, then the data must contain more structure than simply single randomly selected paths. There should be a global structure present within the data. This is in keeping with the discussion on this same dataset by Lee and Verleysen (Lee and Verleysen, 2007). They suggest that the data is

[8]Throughout we estimate the intrinsic dimensionality by taking the average of the following intrinsic dimensionality estimators: Local PCA (Fukunaga and Olsen, 1971), Maximum Likelihood Estimation (Levina and Bickel, 2004) and Packing Numbers (Kégl, 2002)

split into two clusters, and certainly embeddings produced by some algorithms confirm this. However, it is unclear as to what these two clusters represent. At a high level how can the expressions be split into two parts? What is there in the data which lends weight to this argument? This remains unclear in (Lee and Verleysen, 2007) and as such we do not make the same two cluster assumption. Rather we wish to see in an embedding of the Frey faces data some form of multi-level structure. That is, at a high level the expressions exist in separate regions of the embedding, and at a lower level the expressions can be linked together in a meaningful way.

With a definition made as to what constitutes a meaningful embedding of this data we now move on to analyse the performance of the PLML algorithm on this same dataset. To begin, we examine the model level topology to see the structure that has been extracted from the data. If there is no apparent structure or ordering in the data at this level then the embedding produced later on in the algorithm will be unusable. A simple layout of the model level MST produced by the PLML algorithm with model size parameter $k = 7$ is shown in Figure 4.24. A parameter value of $k = 7$ was chosen as the optimal value by eyeballing the results produced by different parameter values and choosing the visually most pleasing. The MST shown in Figure 4.24 was rooted at a random model (in this case a model representing a happy expression). From this root node the models expand out along different paths according to the topology defined by the MST.

Analysing a graph of this form can often be difficult as it is unclear as to what we are looking for. Therefore we keep in mind the goal of the MST in the PLML algorithm: to represent the correct topology of the models. Similar models should be connected and dissimilar models unconnected. We now turn our attention back to Figure 4.24 and see that the different paths taken along

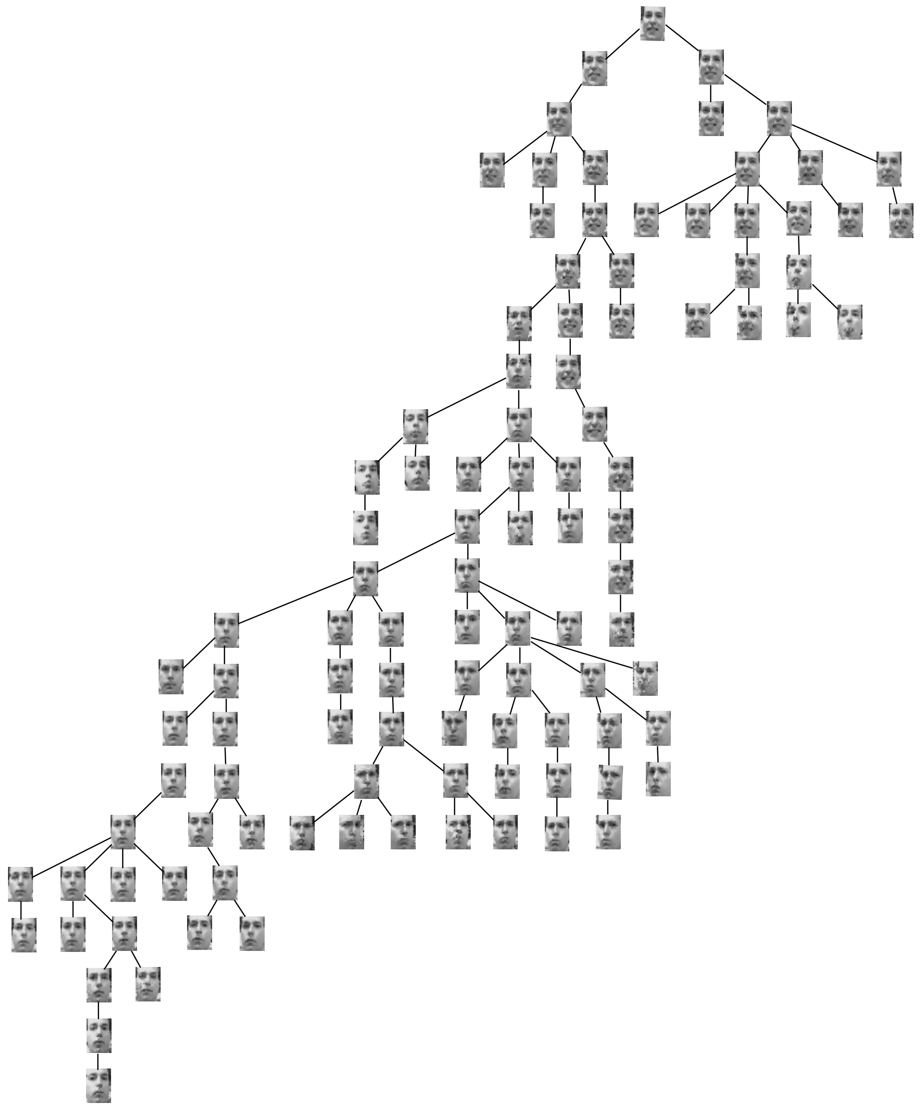


Figure 4.24: The MST topology graph for the Frey faces dataset. Each image is the median face of a model and the graph shows the connectedness of each model according to the MST.

the MST represent different expression changes. Take for example the facial expression at the top of the tree (happy) and one of the facial expressions at the bottom of the tree (neutral or angry). Visually tracing a path through the tree from these two models shows that the expression does interpolate in a meaningful way. A similar example of this is shown in Figure 4.25 where a sequence was taken from the tree and shows the interpolation of facial expression from neutral to angry. What is also noticeable is that the under represented expressions (i.e. sticking tongue out) do not exist on a separate sub-tree of the graph, rather they are the leaf nodes. Conversely the more represented expressions such as neutral or happy are subgraphs (with the example of happy being a subgraph of the top-right of the MST and neutral occupying the lower left hand side).

We now move on to investigate the PLML algorithm's ability to embed the Frey faces dataset into 2-dimensions. As previously discussed this process of embedding into 2-dimensions is not ideal as the intrinsic dimensionality of the data is 7-dimensions. However we would expect some structure to become apparent from this embedding process. The result of embedding the Frey faces dataset is shown in quantised form in Figure 4.26. The quantisation process simplifies the visualisation of the embedding. Since there are too many points to display one image per point we split the embedding space into a grid and display the median face of all the points contained within each grid cell. This means that

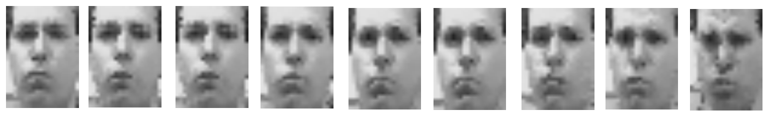


Figure 4.25: A sequence of faces taken from the Frey faces dataset topology graph shown in Figure 4.24. As can be seen the face changes expression from almost neutral to angry. This change can be traced by following a path through the MST.

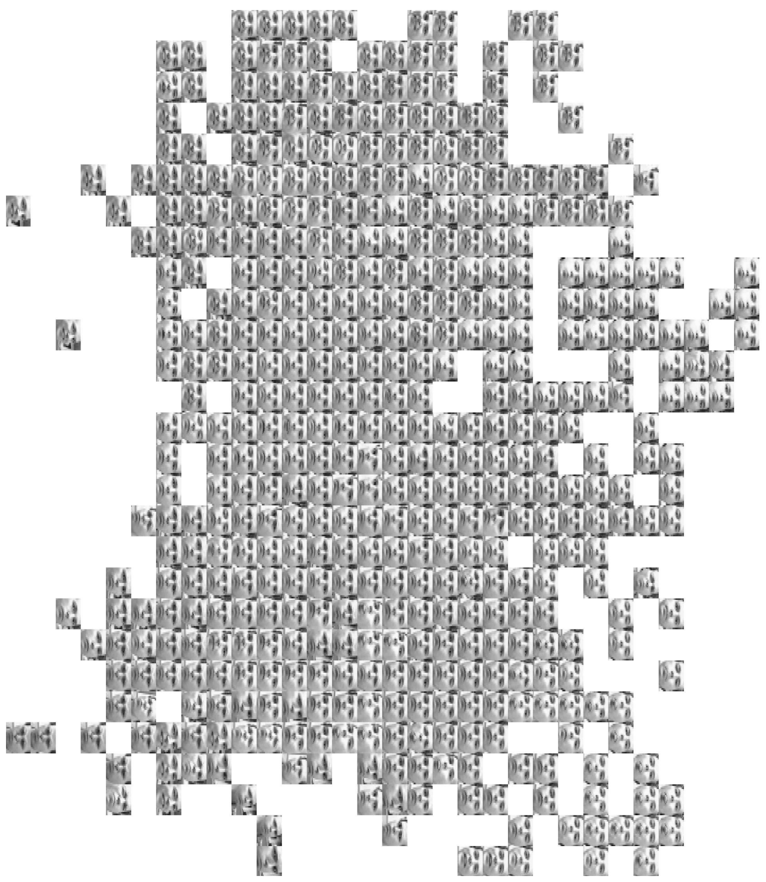


Figure 4.26: Quantised representation of the Frey Faces embedding produced using the PLML Framework.

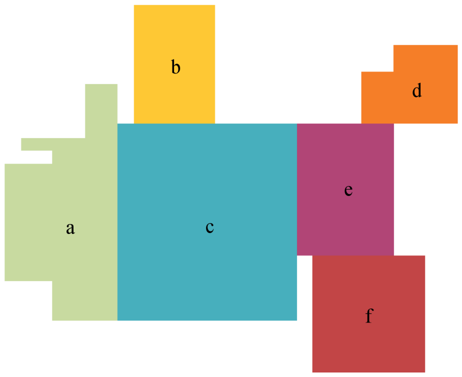


Figure 4.27: The manual segmentation of the Frey faces embedding. Each coloured region represents a different facial expression. The regions were segmented by hand based on the expressions contained in the underlying embedding. See Figure 4.23 for the different expression classes.

an overview of the structure of the manifold can be observed without the need for all data points.

Initially it is difficult to glean any structure or information from the embedding as it does not display a perfect interpolation of expressions across the embedding. However under further examination it becomes clear that the different facial expressions have been segmented into different regions of the manifold. Due to the non-optimal target dimensionality there are some discontinuities in the mapping where some facial expressions 'jump' to different regions but generally different facial expressions reside in different areas of the manifold. To prove this we extend the idea of the quantised mapping to a manual segmentation. Rather than creating a fixed grid and taking the average face we created a hand drawn segmentation of regions (Figure 4.27). The average face of these regions were then taken and can be seen in Figures 4.23 and 4.28. These manual segmentation regions were determined by eye-balling the quantised embedding and determining the expression regions.

From examining this segmentation we can gain a better understanding as to how the PLML algorithm arrived as this embedding. The first thing to note is that the large central region is the neutral expression and then from this the more ‘expressive’ regions branch. This is partly because, as previously discussed, the neutral facial expression is over represented in the data but also because the PLML algorithm manages to successfully learn a meaningful embedding of the manifold. This segmentation of the facial regions into different areas of the embedding shows that PLML has managed to extract useful structure from the data. Whether this is the true structure of the data is unclear as there is no absolute ground truth. However, we can say that the structure displayed in the embedding produced by PLML is useful as it enables us to identify the different facial expression regions in the data.

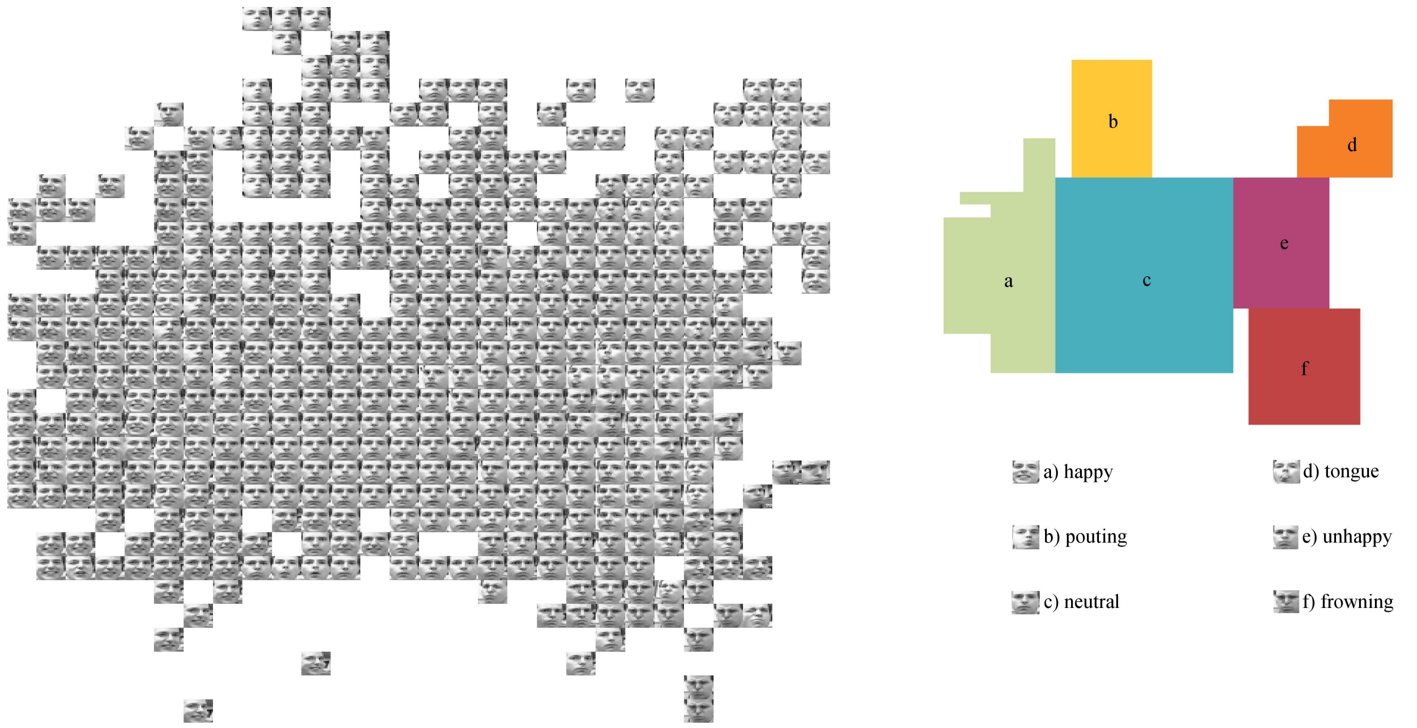


Figure 4.28: Quantised representation of the Frey Faces embedding produced using the PLML Framework (left) along with the manual segmentation (top right) and the mean faces of each manually segmented region (bottom right).

4.4.2 ISOFaces

The ISOFaces dataset (Tenenbaum et al., 2000) is a sequence of images showing a 3-dimensional computer rendered head under different pose and lighting conditions. The pose varies from left to right and up to down. The lighting conditions vary according to the point light's position from left to right. Example images from the data are shown in Figure 4.29.

The intrinsic dimensionality of the dataset is estimated at 6 dimensions and so as with the Frey faces dataset any 2-dimensional embedding will be sub-optimal[9]. This sub-optimality occurs because we are attempting to embed the data set into its non-intrinsic dimension. As discussed with the Frey faces dataset this can introduce distortions and overlaps. This being said since the ISOFaces dataset is artificially generated with well defined parameters it is easier to glean structure from the embeddings as we know we are looking for definite change in pose and lighting. Another real benefit of the data being artificially generated is that the noise level is kept to a minimum.

As with the Frey faces dataset we begin by analysing the MST produced by the PLML algorithm. Figure 4.30 shows the MST produced as a result of performing

[9]The intrinsic dimensionality was estimated using an average of intrinsic dimensionality estimators as with the Frey faces dataset

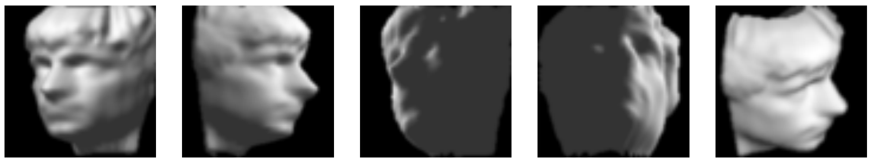


Figure 4.29: Sample images taken from the ISOFaces dataset. The computer rendered head changes left-right and top-down pose as well as lighting conditions.

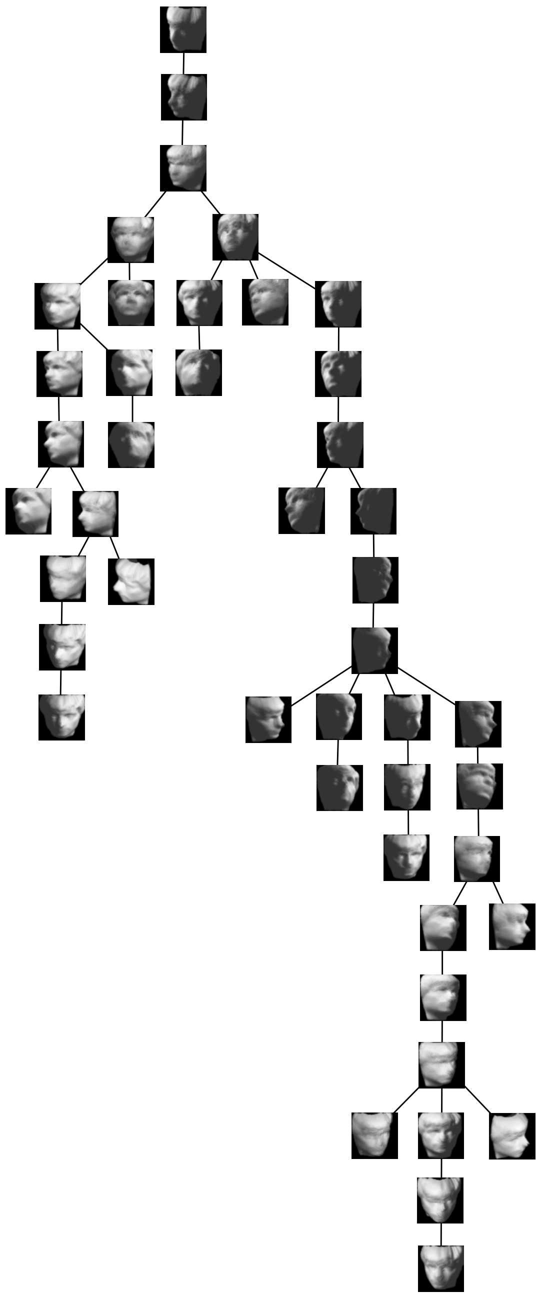


Figure 4.30: The PLML model level MST of the ISOFaces dataset created with $k = 5$.

PLML with $k = 5$ (see footnote[10]). The first noticeable property of the MST is that lighting conditions are separated but the two different lighting conditions (light and dark) do not lie on two distinct sub-graphs of the MST. Rather the MST can be seen to transition from light to dark and then back to light again.

The change in pose variation is more difficult to identify. A close inspection of the MST reveals that if the MST is split at the dark lighting region then the right pose faces generally appear on the subgraph to the left and the left pose faces generally appear on the right subgraph. Within these subgraphs there is up-down pose variation but it is more difficult to identify any structured sub graphs in the same way as the left right poses.

The quantised embedding of the ISOFaces dataset produced by PLML is shown in Figure 4.31. Again the lighting variation is the most immediately noticeable parameterisation of this embedding with the lighting changing from light at the top to dark and the middle and then light again at the bottom. Change in pose can be split into two subsections with the top section (above the dark region) containing the left poses and the bottom section (below the dark region) containing the right pose images. Of these two regions the top (left pose) region is better structured. This distinction is less apparent in the bottom, right pose, region.

Although this embedding does contain some structure it is far from ideal and is not comparable with the structure of embeddings of the same dataset produced by different algorithms (e.g. Tenenbaum et al. (2000); Roweis et al. (2002)). The reasons for this are twofold. First, as discussed when considering the Fishbowl and Helix datasets, PLML struggles when embedding datasets with essential loops. The rotational changes in the pose and the lighting direction suggests

[10]The value of $k = 5$ was reached by eyeballing the embeddings produced across a number of different parameter values and choosing the best result.

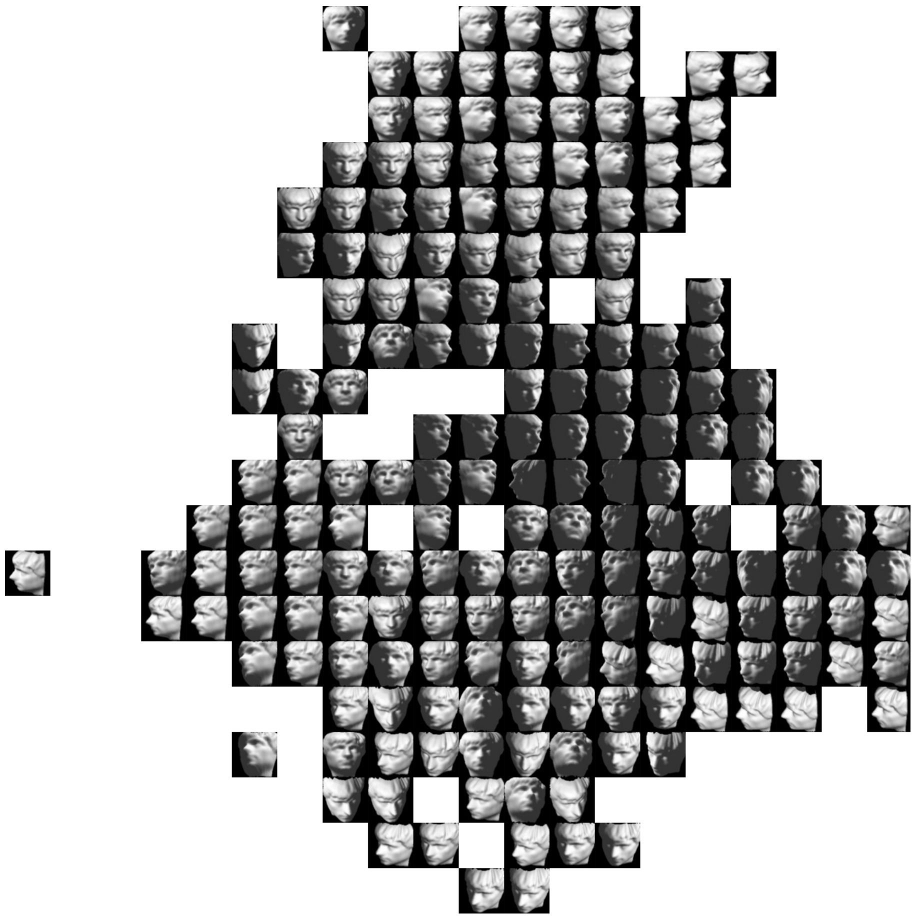


Figure 4.31: The embedding of the ISO Faces dataset as found with PLML $k = 5$. This intrinsic structure of the dataset is difficult to visualise as it is greater than 2-dimensions. However, from this embedding it becomes evident that PLML has managed to distinguish the left-right pose variation as well as the different lighting conditions.

that the ISOFace dataset does contain essential loops and as such the PLML embedding will be sub-optimal. With this in mind it is still useful to know that PLML can still extract some structure from the data. Although PLML does not create a 'perfect' embedding we can still infer structure and meaning about the original data from the produced embedding (i.e. there is change in pose and lighting). Secondly, PLML's performance drops when attempting to embed a dataset into its non-intrinsic dimension. This is shown when considering the Trustworthiness of PLML's embedding of the ISOFaces dataset in 2-dimensions and in 6-dimensions (the intrinsic dimensionality of the dataset). The average Trustworthiness value when embedding into 2-dimensions is $T = 0.851$ whereas when embedding into 6-dimensions this value rises to $T = 0.953$.

The ISOFaces dataset presents a difficult problem for the PLML algorithm as it is not only a dataset being embedded into its non-intrinsic dimension but it is also a difficult manifold with possible essential loops. The embedding shown in Figure 4.31 shows that the PLML algorithm can extract some meaningful structure from this data with the low-dimensional embedding being parameterised mainly by left-right pose and lighting conditions. The embedding into non-intrinsic dimensionality undoubtedly affects the performance of PLML but the MST (Figure 4.30) which is independent of dimensionality does not show as clear a structure as the Frey faces MST (Figure 4.24). This therefore suggests that at the model level PLML is not able to fully recover a good approximation of the manifold with the consequence being that the low-dimensional embedding will not display an optimal structure. The embedding produced however, does display some good structure and does allow us to make inferences about the original data (i.e there is change in pose and lighting positions in a smooth manner).

4.4.3 COIL-20 Object

The COIL-20 (Nene et al., 1996) object data consists of a set of 72 images of a 3-dimensional object rotating around 1 axis. Each image is 32×32 pixels in size and as such this dataset consists of a sparse problem as there are considerably more dimensions than samples (72 samples in 1024-dimensional space). Due to the fact that the object only moves around 1-axis and all other variables (e.g. object size, lighting, etc) are fixed then this data is expected to lie on a 2-dimensional manifold. This manifold is parameterised by the degree of rotation and so equates to a circular manifold embedded within the 1024-dimensional space[11]. This estimation of intrinsic dimensionality is backed up by the average intrinsic dimensionality estimate of 2-dimensions.

The PLML based MST of the COIL-20 object dataset is shown in Figure 4.32. What is immediately noticeable is that the MST is intrinsically 1-dimensional. It has no branches or sub-graphs and simply consists of a linked list of models. The MST also successfully captures the rotation of the object with the models being connected in the correct rotational order. A link could be made between the last model in the MST and the first model to produce a continuous cycle, however

[11]For another discussion on a similar dataset see Chapter 1

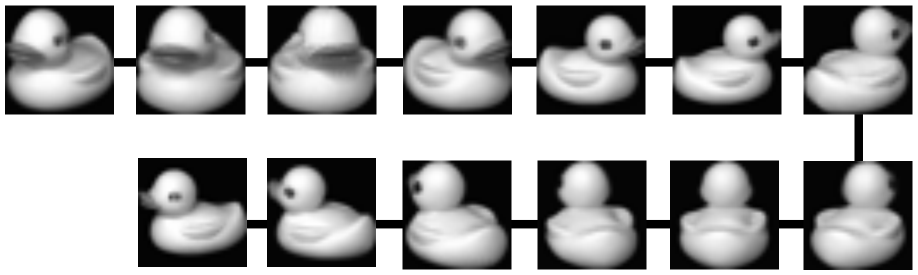


Figure 4.32: The MST of the Coil-20 Object dataset. The MST is intrinsically 1-dimensional, that is it has no branches or possible subgraphs (unlike the MST of the other datasets, e.g. Figure 4.24). The ordering here shows that the MST correctly captures the rotational ordering of the object.

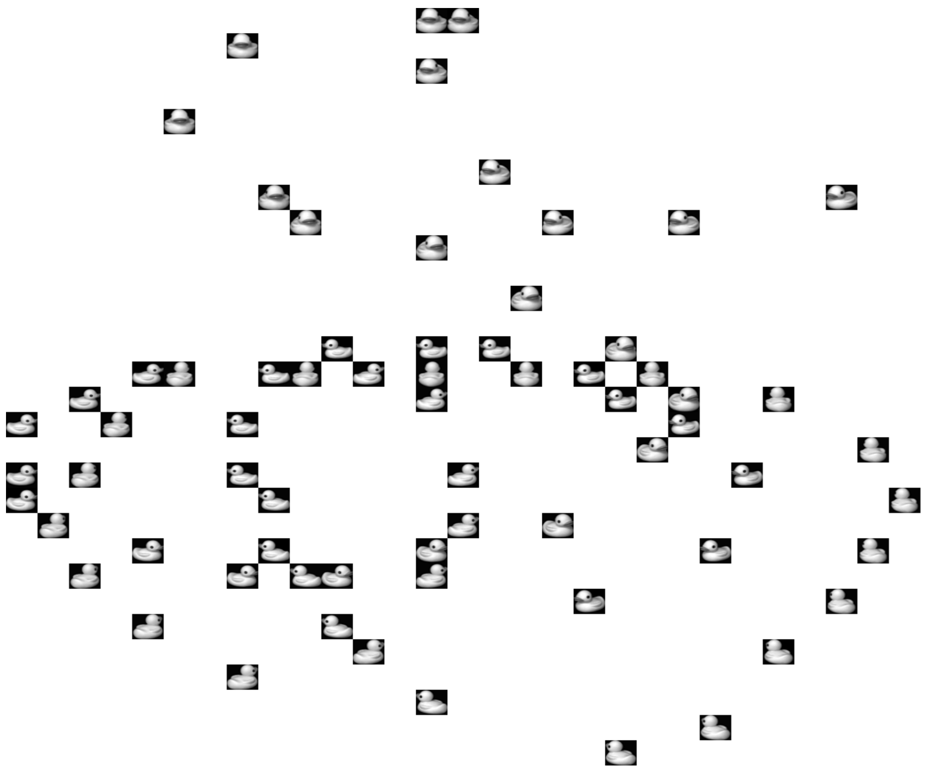


Figure 4.33: The low-dimensional embedding of the COIL-20 dataset produced by PLML with $k = 4$. Notice that the circular structure of the data is not recovered.

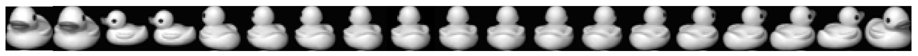


Figure 4.34: The quantised embedding of the COIL-20 dataset produced using PLML. Notice that the rotation of the object is captured in the 1-dimensional embedding whereas this structure is less apparent in the 2-dimensional embedding (Figure 4.33).

this link does not exist. As such this dataset resembles the artificial Helix dataset discussed in Section 4.2 and we expect PLML to struggle to embed this data into 2-dimensions. This point is indicated by the embedding shown in Figure 4.33. This embedding is not quantised as with the Frey faces and ISOFaces dataset. Rather every low-dimensional image is shown. The parameter value of $k = 4$ was chosen as this is a sparse problem and the experimentation in Section 4.3.3.1 showed that for sparse problems a small neighbourhood size should be chosen. The embedding in Figure 4.33 does exhibit some structure. We could trace a path between the different data points and glean that the object rotates around one axis. So the embedding is not unstructured but it does not map to a circle. With the knowledge we have as to the performance of PLML on datasets with essential loops this is not surprising.

Returning to the similarities between this dataset and the Helix coil dataset, we can view this dataset as a 1-dimensional curve and therefore attempt to embed it into 1-dimensional space. The quantised result of embedding the COIL-20 dataset into 1-dimensional space using PLML is shown in Figure 4.34[12]. By embedding the data into 1-dimensional space PLML has managed to capture the structure of the data and clearly shows that the embedding is parameterised by the degree of rotation of the object. So as with the Helix coil dataset PLML has produced a more meaningful and accurate embedding by embedding the circular manifold into 1-dimensional space rather than 2-dimensional space.

[12]We return to showing a quantised embedding to be able to better show the structure within the embedding

4.5 Computational Complexity

The computational complexity of a manifold learning algorithm is determined by the properties of the dataset (e.g. the number of data points, n, the dimensionality of the input data, p) as well as the properties of the algorithm (e.g. the parameter size k, target dimension d and the number of local models c).

PLML has three main computational steps, the k-means clustering step, the creation of the MST and the traversal of the MST. The k-means clustering step can be computed in $O(nk)$ given that p and k are fixed. The MST can be computed in $O(n^2 + n)$ where n^2 is the number of edges in the original graph. The MST traversal contains a number of iterations with matrix multiplication occurring at each iteration. Experimental analysis shows that the number of iterations, on average, is $2c$ as each model is usually visited twice during the traversal process. Here, $c = \lfloor n/k \rfloor$. Matrix multiplication has computational complexity of $O(p^3)$, so given that we perform matrix multiplication $2c$ times, the computational complexity for this step becomes $O(2cp^3)$.

The computational complexity of the PLML algorithm therefore is

$$O(nk) + O(n^2 + n) + O(2cp^3) \tag{4.6}$$

which reduces to

$$O(2cp^3) \tag{4.7}$$

The computational complexity of the other four manifold learning algorithms we are considering are as follows. ISO: $O(n^3)$, LLE: $O(rn^2)$, LTSA: $O(rn^2)$, PCA: $O(p^3)$, where r is the ratio of nonzero elements in a sparse matrix to the total number of elements. In the case of data where $p < n$ then PLML has a better computational complexity performance than ISO, LLE and LTSA. When $n \ll p$ the computational performance of PLML drops and ISO, LLE and LTSA will outperform the developed method. As such it is worth noting that in cases where the dimensionality is far greater than the number of data points then PCA should be used as a pre-processing step to PLML to improve the run-time of the PLML algorithm. The computational complexity of PCA is low enough not to add a needless computational burden.

4.6 Conclusions

In this chapter we have performed both a high-level and detailed analysis of the performance of the PLML algorithm on artificially generated data. We have also analysed the performance of PLML on high-dimensional image data that exhibit interesting and difficult manifold structure. From these experiments we can draw the following conclusions as to the performance and behaviour of the PLML algorithm:

1. As the sparsity of the data increases the performance of the PLML algorithm decreases at a global and local scale. However, the performance at a local scale can be improved by reducing the neighbourhood size parameter k.

2. Under high noise PLML fails to recover a global and locally stable embedding. As with increased sparsity this problem can be slightly alleviated by changing the neighbourhood size parameter. Unlike the sparse case, k needs to be increased as noise increases to compensate for the effect of noise.

3. With dense, low noise data, the neighbourhood size parameter does not play as much of a role on the quality of the embedding and so parameter choice does not become such an important issue. This is not the case for the other global alignment technique, LTSA. Parameter choice plays a major part in the performance of LTSA and the incorrect parameter choice can have drastic effects on the performance.

4. PLML struggles to embed datasets with essential loops. Circular or spherical manifolds will have to be cut to embed them into the low-dimensional space. This can cause problems as shown in the Helix dataset and also the Coil-20 image dataset. This can be overcome in the circular case by embedding the data into 1-dimension.

5. We have also shown that PLML is able to extract meaningful embeddings from difficult image datasets. These embeddings allow us to draw conclusions as to the nature of the high-dimensional data and so PLML is well suited as an exploratory data analysis tool.

6. When $p < n$ the computational complexity of PLML is less than that of Isomap, LLE and LTSA. However when $n \ll p$ then the computational complexity of PLML increases. As such PCA could be used as a pre-processing step to improve the run-time of the PLML algorithm.

The results in this chapter show that modelling the manifold in a piecewise-linear manner, by using PLML, can achieve significantly improved results over existing manifold learning methods. This is especially the case when the data is densely sampled with low to medium noise. Even when the density decreases and noise increases, PLML is still able to degrade gracefully and produce meaningful results.

"To learn something new,

take the path that you took yesterday."

John Burroughs (1837-1921)

5 A Generalised Solution to the Out-of-Sample Extension Problem[1]

The out-of-sample extension problem in manifold learning is concerned with embedding novel high-dimensional data points into a previously learnt embedding. The brute force approach is to append the new data point to the original high-dimensional data and then re-learn the low-dimensional embedding to find the new data point's low-dimensional position. However, this can often be computational unfeasible and is not the most elegant solution to the problem.

The out-of-sample extension enables manifold learning to be used in many application areas. For example, in a classification task we may have a set of high-dimensional training samples which we map, using manifold learning, to

[1]Harry Strange and Reyer Zwiggelaar. A Generalised Solution to the Out-of-Sample Extension Problem in Manifold Learning. In Proceedings of AAAI-11, San Francisco, CA. August 2011.

a low-dimensional space. We then have a set of test samples which we wish to embed into the low-dimensional space and classify according to the previously learnt training patterns. The out-of-sample extension can be used to quickly map these test samples to the low-dimensional space without re-learning the entire training set at the same time. This can be particularly useful if we are not classifying all the test samples at once but rather they arrive and are classified in an incremental manner.

A related problem to the out-of-sample extension is that of incremental learning (e.g. Incremental Isomap (Law and Jain, 2006), Co-ordinate Propagation (Xiang et al., 2009)). Manifold learning algorithms traditionally learn the dataset in one single run (batch learning). This restriction means that manifold learning can not traditionally be applied to sequential data. Incremental learning seeks to overcome this problem. The incremental version of Isomap (Law and Jain, 2006) incrementally builds the low-dimensional embedding by appending new samples to the low-dimensional space in such a way that new data points hold more significance to the embedding than those previously learnt (with the old data points gradually losing their significance). This then enables the algorithm to handle changes in the manifold's characteristics as the data is learnt. Although at first sight incremental learning may seem similar to the out-of-sample extension problem there are core differences. The most significant of these being that the addition of new data points using the out-of-sample extension does not cause a re-learning of the parameterisation of the manifold. That is, the information used to embed the new samples is not updated with every new sample. For incremental manifold learning there is a continuous update which increases the computational cost.

Incremental learning is closely related to manifold extrapolation, where the previously learnt manifold is extended as new points are added to the embedding (e.g. (Chin and Suter, 2008)). Incremental learning allows for such extrapolation whereas the out-of-sample extension will be less robust to embedding new points outside of the previously learnt manifold. One of the core assumptions of the existing solutions to the out-of-sample extension is that the new sample lies 'within' the previously learnt manifold. This is due to the fact that properties of the existing manifold are used to find the new sample's low-dimensional location. If this new sample lies far outside of the existing manifold then there will not be enough information to correctly position it in the low-dimensional space. As such, it is worth remembering that out-of-sample extension methods may not be able to perform manifold extrapolation unless the new points lie close to the original manifold's samples.

Many of the existing solutions to the out-of-sample extension problem are algorithm specific (i.e. they only work with certain manifold learning algorithms). Some manifold learning algorithms come ready equipped with solutions to the out-of-sample extension problem (e.g. Manifold Charting (Brand, 2003) and KernelPCA (Scholkopf et al., 1998)). Other algorithms have been explicitly extended to attempt to cope with the out-of-sample extension problem (e.g. Isomap, LLE, Eigenmaps and MDS (Bengio et al., 2003)). These solutions are discussed in more detail in Section 5.2 but for now it is worth noting that although these algorithms work well at embedding new data points into previously learnt embeddings little work has been carried out to produce a generalised solution to this problem.

In this chapter we present a generalised solution to the out-of-sample extension problem. Our developed method, GOoSE (Generalised Out-of-Sample Extension), works by learning the local geometric transform that occurs as a result of manifold learning. This local transform describes how a local neighbourhood is distorted and changed as a result of manifold learning. By learning the geometric change in an unseen sample's neighbourhood we can find the sample's position in the low-dimensional space. Our algorithm is simple to understand and fast to compute. It also out-performs existing algorithm specific out-of-sample methods. We show that it can be applied to any manifold learning algorithm as long as we have access to the original high-dimensional data and the low-dimensional embedding.

The remainder of this chapter is structured as follows. We begin in Section 5.1 by outlining the mathematical foundations that underpin the out-of-sample extension problem. In this section we phrase the out-of-sample extension within the algorithm of manifold learning so as to better understand how it fits into the research of the previous chapters. Section 5.2 provides an overview of existing solutions to the out-of-sample extension problem. We present a generalised solution to the out-of-sample extension problem in Section 5.3 and compare our generalised solution with other algorithm specific solutions in Section 5.4. In this section we also discuss the general performance of the developed method. Finally, we draw conclusions in Section 5.5.

5.1 Problem Description

The out-of-sample extension problem can be defined as follows: given a set of observations $\mathbf{X} \in \mathbb{R}^p$ and its low-dimensional embedding $\mathbf{Y} \in \mathbb{R}^q$ found as a result of manifold learning, we wish to find the low-dimensional position in $\mathbb{R}^q$ of a novel observation $x \notin \mathbf{X}, x \in \mathbb{R}^p$ without using the entire dataset. We expand upon this definition in more detail below.

Given two real vector spaces $\mathbb{R}^p$ and $\mathbb{R}^q$, and a manifold embedded in each space $\mathcal{I} \in \mathbb{R}^p$, $\mathcal{J} \in \mathbb{R}^q$, we wish to find a function that maps between the two spaces $f : \mathbb{R}^p \to \mathbb{R}^q$. For the problem of manifold learning $\mathcal{I}$ is known and f is used to learn the low-dimensional manifold $\mathcal{J}$. But consider the case where both $\mathcal{I}$ and $\mathcal{J}$ are known, the problem now becomes learning f so that new samples from $\mathcal{I}$ can be embedded into $\mathcal{J}$ without re-learning the entire manifold. This is the core of the out-of-sample problem and seeks to embed novel samples into a previously learnt embedding.

Given a sample $x \in \mathbb{R}^p$, $x \subset \mathcal{I}$ and a function $f : \mathbb{R}^p \rightarrow \mathbb{R}^q$ such that f also maps between manifolds $f : \mathcal{I} \mapsto \mathcal{J}$, then $y = f(x)$ given $y \in \mathbb{R}^q$ and $y \subset \mathcal{J}$. This means that the function f not only maps between spaces but also maps between the manifolds such that if the sample lies on the manifold in the high-dimensional space it will also lie on the manifold in the low-dimensional space.

Manifold learning provides us with a setting within which the above assumption holds. Given a set of observations $\mathbf{X} = \{x_i\}_{i=1}^n \in \mathbb{R}^p$ where $\mathbf{X} \subset \mathcal{I}$ then manifold learning will attempt to find the set of low-dimensional observations $\mathbf{Y} = \{y_i\}_{i=1}^n \in \mathbb{R}^q$ such that $\mathbf{Y} \subset \mathcal{J}$. Since $\mathcal{J}$ is assumed to be an embedding of $\mathcal{I}$ in the low-dimensional space (i.e. $\mathcal{I} \hookrightarrow \mathcal{J}$) then we can say that $\mathbf{Y}$ is an embedding of $\mathbf{X}$. This means that the above function can be expressed in terms of $\mathbf{X}$ and $\mathbf{Y}$ such that $f : \mathbf{X} \mapsto \mathbf{Y}$.

The out-of-sample problem can now be defined in terms of $\mathbf{X}$ and $\mathbf{Y}$. Given a novel observation $x \in \mathbb{R}^p$ and $x \subset \mathcal{I}$ but $x \notin \mathbf{X}$ then we wish to find its low-dimensional representation $y \in \mathbb{R}^p$ where $y \subset \mathcal{J}$. This is done by learning the function $f : \mathbf{X} \mapsto \mathbf{Y}$ that maps from the high-dimensional space to the low-dimensional space. Some manifold learning algorithms provide this function (e.g. Kernel PCA (Scholkopf et al., 1998) and Manifold Charting (Brand, 2003)) but we wish to find a generalised function given that we know $\mathbf{X}$ and $\mathbf{Y}$.

Given that we now have a proper definition of the out-of-sample extension problem we can move on to discuss existing solutions to this problem.

5.2 Existing Solutions

5.2.1 Principal Components Analysis

Many algorithm specific solutions exist to solve the out-of-sample extension problem. Most of these utilise the information found during the manifold learning process. The simplest example is that of Principal Components Analysis (PCA) (Hotelling, 1933). As shown previously (see Chapter 2), PCA seeks to find the set of basis vectors that span the hyperplane of maximum variance, the solution to which can be found by

$$\Lambda \mathbf{V} = \mathbf{C_X} \mathbf{V} \tag{5.1}$$

where $\mathbf{C_X}$ is the covariance matrix of $\mathbf{X}$, Λ is the set of eigenvalues of $\mathbf{C_X}$ and $\mathbf{V}$ are their associated eigenvectors. To then reduce the dimensionality of $\mathbf{X}$ we would project $\mathbf{X}$ onto the top q eigenvectors in $\mathbf{V}$. Therefore we can state that the low-dimensional representation of $\mathbf{X}$ found by PCA is $\mathbf{Y} = \mathbf{X}\mathbf{V}_{1\ldots q}$.

The matrix $\mathbf{V}$ contains all the information about our low-dimensional space as described by PCA. So given a new observation, x^*, we can embed it into this same low-dimensional space by $y^* = x^*\mathbf{V}_{1\ldots q}$. By retaining the eigenvectors found in Eq. 5.1 we can then embed new data points into the low-dimensional space by projecting them onto these basis vectors.

This simple method provides the basis for most other algorithm specific solutions to the out-of-sample extension problem. Properties learnt as a result of manifold learning are retained and then applied to novel samples to embed them into the previously learnt low-dimensional embedding. The two most relevant to our work are that of Bengio et al. (Bengio et al., 2003) and Yang et al. (Yang et al., 2010).

5.2.2 Bengio et al.

The first attempt to provide an out-of-sample extension for algorithms that do not naturally contain one was made by Bengio et al. in 2003 (Bengio et al., 2003). The algorithm presented in (Bengio et al., 2003) extended LLE (Roweis and Saul, 2000), Isomap (Tenenbaum et al., 2000), MDS (Cox and Cox, 2001), Eigenmaps (Belkin and Niyogi, 2002) and Spectral Clustering (Weiss, 1999; Ng et al., 2002) to handle the out-of-sample extension problem. These techniques are extended by phrasing them within a unified kernel algorithm. Within this algorithm the algorithms are seen as learning eigenfunctions of a kernel (Bengio et al., 2003).

To begin with, an $n \times n$ kernel matrix is formed, $\mathbf{K}$. This matrix is built using a two-argument kernel function $\phi(.,.)$ which produces $\mathbf{K}$ such that $\mathbf{K}_{ij} = \phi(x_i, x_j)$. This matrix can then be normalised to obtain $\bar{\mathbf{K}}$. The largest eigenvalues λ and eigenvectors $\mathbf{V}$ (with a single eigenvector, $v_i \in \mathbf{V}$) of $\bar{\mathbf{K}}$ are then found with the restriction that only positive eigenvalues should be considered. The embedding of each point x_i is then y_i with y_{ij} being the i-th element of the j-th principal eigenvector, v_j, of $\bar{\mathbf{K}}$.

This solution is then extended to the specific cases of LLE, Isomap, MDS, Eigenmaps and Spectral Clustering. This extension is based on a continuous kernel function which can be used to generalise an embedding to a new data sample[2]. A separate kernel function then needs to be learnt for each manifold

[2]We omit the full details of this step and refer the reader to Section 3 and Section 4 of (Bengio et al., 2003) for full details.

learning algorithm. This design of data dependent kernel matrices for each manifold learning algorithm is a non-trivial task (Yang et al., 2010). However, once an appropriate kernel matrix, $\mathbf{K}$, has been defined a new point can be embedded according to the Kernel PCA (Scholkopf et al., 1998) projection with kernel $\mathbf{K}$.

Although this method enables LLE, Isomap, MDS, Eigenmaps and Spectral Clustering to include new data points, it does so by learning very specific kernel functions. This limits the approach to be used for other manifold learning algorithms as it is unfeasible to learn separate kernel matrices for each new manifold learning algorithm.

5.2.3 Yang et al.

A similar attempt to phrase the manifold learning problem within a common algorithm was made by Yang et al. in (Yang et al., 2010). Rather than learning a continuous kernel function as in (Bengio et al., 2003), Yang et al. seek to employ local regression models to learn the manifold structure. Within this work a new algorithm for manifold learning is presented alongside a generalised out-of-sample extension. Throughout this discussion we focus solely on the out-of-sample extension. Given a new point, x^*, its low dimensional embedding y^* is estimated as

$$y = \phi(\mathbf{W})^T \phi(x^*) + b \tag{5.2}$$

where ϕ is the kernel function mapping data into Hilbert space, $\mathbf{W}$ is the projection matrix from Hilbert space to the low-dimensional space and b is a bias term. This can then be computed by

$$y^* = \mathbf{Y}^T(\mathbf{HKH} + \gamma\mathbf{I})^{-1}\mathbf{HK}_x + \frac{1}{n}\mathbf{Y}^T 1_n - \frac{1}{n}\mathbf{Y}^T(\mathbf{HKH} + \gamma\mathbf{I})^{-1}\mathbf{HK}1_n \quad (5.3)$$

where $\mathbf{H}$ is the global centring matrix, γ is a regularisation parameter (with $\gamma > 0$) and $\mathbf{K}$ is a matrix containing the result of the Kernel product such that $\mathbf{K}_{x^*i} = \phi(x^*)^T\phi(x_i)$. As such $\mathbf{K}$ contains the Kernel product of the new datapoint with all data points in the training set.

This method therefore seeks to generalise Bengio et al.'s solution by using any positive semidefinite Kernel and adding a regularisation parameter, γ, to control the bias. However, the presence of this regularisation parameter significantly complicates the out-of-sample process as there is no guidance on what value γ should take. The choice of Kernel (e.g. Linear, RBF, etc) will also affect the low-dimensional embedding and little is discussed in (Yang et al., 2010) as to what Kernel should be used. In addition, since the out-of-sample calculation involves all previously learnt data, the computation could potentially be expensive.

With these limitations to existing solutions the need for a simple, generalised solution becomes apparent. In the next section we outline the developed method which seeks to obtain a generalised solution to the out-of-sample extension problem.

5.3 A Generalised Solution

The basic premise of our algorithm is to find the transformation that matches an unseen samples's neighbourhood in the high-dimensional space to its representation in the low-dimensional space. This transformation is an approximation of running the manifold learning technique on the given sample.

Given the original set of points $\mathbf{X} = \{x_i\}_{i=1}^{n} \in \mathbb{R}^p$, where $\mathbf{X} \subset \mathcal{I}$, and their low-dimensional representation $\mathbf{Y} = \{y_i\}_{i=1}^{n} \in \mathbb{R}^q$, where $\mathbf{Y} \subset \mathcal{J}$, we wish to find the low-dimensional approximation $y^* \subset \mathcal{J} \in \mathbb{R}^q$ of a previously unseen sample $x^* \subset \mathcal{I} \in \mathbb{R}^p$ given $x^* \notin \mathbf{X}$. As described in Section 5.1 we assume that there exists a function, f, that maps not only between the two real vector spaces, $\mathbb{R}^p$ and $\mathbb{R}^q$, but also between the two manifolds such that $f : \mathcal{I} \mapsto \mathcal{J}$. Since $\mathbf{Y}$ is an embedding of $\mathbf{X}$ in $\mathbb{R}^q$, we can say that f also maps between the two sets of data, $f : \mathbf{X} \mapsto \mathbf{Y}$. We can re-write this as

$$\mathbf{Y} = f(\mathbf{X}) \tag{5.4}$$

We seek to learn this function to extend the mapping to new data points. However, in all but the linear case this function f will be non-trivial to learn. For the methods presented by (Bengio et al., 2003) and (Yang et al., 2010) this function is equivalent to learning the kernel function that re-creates $\mathbf{X}$ in $\mathbb{R}^q$ as $\mathbf{Y}$. Our approach differs as it is based on learning the local function that builds a neighbourhood of $\mathbf{X}$ in $\mathbf{Y}$. That is, for the i-th data point we express an individual local function

$$y_i = f_i(x_i) \tag{5.5}$$

This is in contrast with the methods in (Bengio et al., 2003) and (Yang et al., 2010) where the global kernel matrix is used as the main mapping function between the high and low-dimensional spaces. For the generalised solution we can express this local function as $y^* = f^*(x^*)$, where f^* is learnt according to the existing mapping $\mathbf{X} \mapsto \mathbf{Y}$.

To formulate a solution to this local mapping function we return to one of the central assumptions in manifold learning. Given a manifold, $\mathcal{I}$, we assume that the manifold $\mathcal{I}$ is a C^∞ manifold and at a local scale $\mathcal{I}$ is linear. This is based on the work by (Roweis and Saul, 2000) and further by (Zhang and Zha, 2004; Brand, 2003) which states that given a subset $\mathcal{K} \subset \mathcal{I}$ represented by a k-neighbourhood of $\mathbf{X}$, then all but the top q eigenvalues associated with the basis vectors of this neighbourhood will be 0. Thus, the manifold at a local scale is linear. Following on from this assumption we can state that the local function, f_i, can be expressed in terms of a linear transformation. This linear transformation will seek to recover the change in neighbourhood geometry around a data point that occurs as a result of manifold learning. Once we have learnt this transformation we can use it to approximate the new data point's position in the low-dimensional embedding. As such we can express Eq. 5.5 as

$$y^* = \mathbf{A}\mathbf{V}x^* \tag{5.6}$$

where $\mathbf{V}$ is the projection matrix from $\mathbb{R}^p$ to $\mathbb{R}^q$ and $\mathbf{A}$ is a similarity transform representing the change in local geometry that occurs as a result of manifold learning.

Our generalised solution to the out-of-sample extension problem is achieved through the following steps:

1. Find the k-nearest neighbours of x^* in $\mathbf{X}$
2. Find the projection matrix to project the nearest neighbours from $\mathbb{R}^p$ to $\mathbb{R}^q$
3. Calculate the transformation matrix measuring the change between the

actual embedding of the neighbourhood of x^* and the projected embedding as found in Step 2

4. Apply the projection matrix (Step 2) and transformation matrix (Step 3) to x^* to find its low-dimensional representation y^*.

We expand upon each of these steps in the text below and then provide a pseudocode implementation of the developed approach.

5.3.1 Step 1: Find The Local Neighbourhood

The first step is concerned with finding the local neighbourhood of the new sample, x^*, within the existing data $\mathbf{X}$. This neighbourhood will give us the position of x^* relative to existing points in the high and low-dimensional space. By recovering this neighbourhood in the low-dimensional space we can approximate the position of y^* relative to its nearest neighbours.

The k-nearest neighbours can be found using a variety of different approaches (e.g. (Kleinberg, 1997; Nene and Nayar, 1997; Garcia et al., 2008)) and the central idea is to find the datapoints in a test set, $\mathbf{X}$, that are closest to the query point, x^*, according to the Euclidean distance[3]. The value of k should be an integer with $k \ll n$. We investigate the effect of differing sizes of k in Section 5.4.

The output of our nearest neighbour search is a $k \times q$ matrix, $\mathbf{X}_{\mathcal{N}}$, which contains as rows the k-nearest data points of x^* in $\mathbf{X}$.

[3]An interesting extension to the developed approach would be to try different metrics. However since most manifold learning algorithms use the Euclidean metric we focus solely on the Euclidean distance

5.3.2 Step 2: Calculate The Projection Matrix

Once the k-nearest neighbours of x^*, $\mathbf{X}_{\mathcal{N}}$, have been found we are able to project x^* into q-dimensional space by projecting onto the top q eigenvectors of $\mathbf{X}_{\mathcal{N}}$. This will give us y^* which is the low-dimensional representation of x^*.

To find the projection matrix, $\mathbf{V}$, we perform Principal Components Analysis on $\mathbf{X}_{\mathcal{N}}$. To do this we first zero mean the data $\mathbf{X}_N$ and form $\mathbf{C}$ which is the covariance matrix of $\mathbf{X}_{\mathcal{N}}$. The principal components can then be found by

$$\mathbf{V}\Lambda = \mathbf{C}\mathbf{V} \tag{5.7}$$

where $\mathbf{V}$ is the matrix containing as columns the eigenvectors of $\mathbf{C}$ ordered according to their eigenvalues Λ. To find the projection matrix into q-dimensional space we take the top q columns of $\mathbf{V}$ represented as $\mathbf{V}_{1\dots q}$. The low-dimensional projection can then be given by $y^* = x^*\mathbf{V}_{1\dots q}$ and also $\mathbf{Y}'_{\mathcal{N}} = \mathbf{X}_{\mathcal{N}}\mathbf{V}_{1\dots q}$. It is worth noting that the projected neighbourhood is denoted as $\mathbf{Y}'_{\mathcal{N}}$ to distinguish it from the same neighbourhood in the low-dimensional embedding $\mathbf{Y}_{\mathcal{N}}$. For simplicity of reading, throughout the remainder of this discussion we denote the the top q eigenvectors simply as $\mathbf{V}$.

Although $y^* \in \mathbb{R}^q$ it is not in the same co-ordinate system as $\mathbf{Y}$ so Step 3 is concerned with aligning y^* into the co-ordinate system of $\mathbf{Y}$.

5.3.3 Step 3: Calculate The Transformation Matrix

The central step of the developed approach is to compute the local transformation that occurs as a result of manifold learning. As described above, the new sample x^* has been projected into low-dimensional space, however, its low-dimensional representation y^* is not necessarily in the same co-ordinate system as $\mathbf{Y}$. So the goal of this step is to align y^* so as to fit into the same co-ordinate system as the embedding $\mathbf{Y}$.

This is achieved by aligning the projected neighbourhood, $\mathbf{Y}'_{\mathcal{N}}$, with the actual embedding of this neighbourhood $\mathbf{Y}_{\mathcal{N}}$. This alignment can be learnt as a translation, rotation and scale. Once learnt, this similarity transformation can be applied to y^* to find its correct location in $\mathbb{R}^q$. Figure 5.1 shows this process. The position of y^* is known relative to the projected neighbourhood $\mathbf{Y}'_{\mathcal{N}}$ and we wish to find its position relative to the actual neighbourhood $\mathbf{Y}_{\mathcal{N}}$. This can be achieved by applying to y^* the translational, rotational and scale change that occurs between $\mathbf{Y}'_{\mathcal{N}}$ and $\mathbf{Y}_{\mathcal{N}}$. This is represented in Figure 5.1 as the similarity transformation matrix $\mathbf{A}$ plus the translation vector v.

The translation vector is computed first. It is found by calculating the difference in means between the two low-dimensional neighbourhoods:

$$v = \bar{\mathbf{Y}}_{\mathcal{N}} - \bar{\mathbf{Y}}'_{\mathcal{N}} \tag{5.8}$$

This then ensures that $\| \mathbf{Y}'_{\mathcal{N}} - \mathbf{Y}_{\mathcal{N}} \| = 0$ which is an important property,

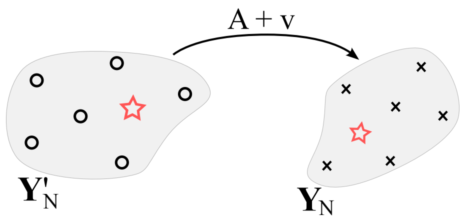


Figure 5.1: To find the position of the new data point (denoted as red star) we seek to find the similarity transform, $\mathbf{A}$, between the two different neighbourhoods (here represented as $\mathbf{Y}'_{\mathbf{N}}$ for the neighbourhood as a result of projection and $\mathbf{Y}_{\mathbf{N}}$ as the real neighbourhood)

because when applying the similarity transform we need to ensure that the difference in means between the two neighbourhoods is zero. This means that no translational distortions occur as a result of computing the similarity transform matrix.

The similarity transform matrix, **A**, can be represented by a separate scale component, **B**, and rotation component, **R**. These two components are learnt separately with the rotation component being learnt first.

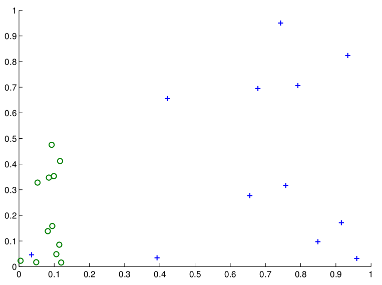
The rotational difference between $\mathbf{Y}'_{\mathcal{N}}$ and $\mathbf{Y}_{\mathcal{N}}$ can be found by using a method from statistical shape theory (Gower, 1975; Wang and Mahadevan, 2008). The rotation matrix can be found by computing the Singular Value Decomposition (SVD) of the matrix $\mathbf{Y}'^{T}_{\mathcal{N}}\mathbf{Y}_{\mathcal{N}}$. The SVD is given by

$$\mathbf{Y}'^{T}_{\mathcal{N}}\mathbf{Y}_{\mathcal{N}} = \mathbf{U}\Sigma\mathbf{V}^{T} \tag{5.9}$$

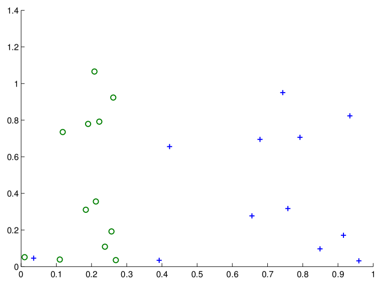
where Σ contains the singular values of $\mathbf{Y}'^{T}_{\mathcal{N}}\mathbf{Y}_{\mathcal{N}}$ as its diagonal. The matrices **U** and **V** contain the left and right singular vectors according to the singular values in Σ. The rotation matrix is then given by

$$\mathbf{R} = \mathbf{U}\mathbf{V}^{T} \tag{5.10}$$

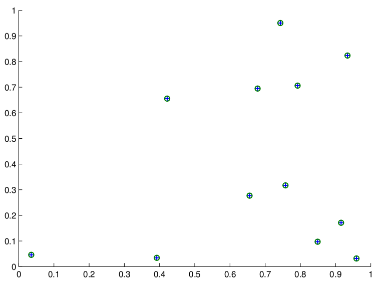
The above process also gives us a scale change value, $b = \text{tr}(\Sigma)$. This value however only represents the isotropic scale (i.e. the value is scalar meaning the scale occurs equally for each axis). The transformation that occurs as a result of manifold learning can often introduce non-isomorphic scale for a local neighbourhood. Therefore we need a scaling matrix that can handle the different amounts of scale for each axis. We define the non-isomorphic scale matrix as



(a) No scale



(b) Isomorphic scale



(c) Non-isomorphic scale

Figure 5.2: The effect of different types of scaling. (a) shows the two sets of points and (b) shows the result of isomorphic scaling to the set of circle points. As can be seen there is still a large variation in scale between the two sets. (c) shows the use of non-isomorphic scaling where different scale values are applied to each axis resulting in a perfect match between the two point sets.

$$\mathbf{B} = \begin{bmatrix} \frac{\max(\mathbf{Y}^1_{\mathcal{N}})-\min(\mathbf{Y}^1_{\mathcal{N}})}{\max(\mathbf{Y}'^1_{\mathcal{N}})-\min(\mathbf{Y}'^1_{\mathcal{N}})} & \cdots & \cdots \\ \vdots & \ddots & \vdots \\ \cdots & \cdots & \frac{\max(\mathbf{Y}^q_{\mathcal{N}})-\min(\mathbf{Y}^q_{\mathcal{N}})}{\max(\mathbf{Y}'^q_{\mathcal{N}})-\min(\mathbf{Y}'^q_{\mathcal{N}})} \end{bmatrix} \tag{5.11}$$

where $\mathbf{Y}^1_{\mathcal{N}}$ represents the column vector containing all samples along the first dimension of $\mathbf{Y}_{\mathcal{N}}$ and $\mathbf{Y}^q_{\mathcal{N}}$ represents the column vector containing all samples along the q-th dimension. This scale matrix measures the difference in size along each dimension between the two neighbourhoods. By multiplying the rotated representation of $\mathbf{Y}'_{\mathcal{N}}$ with $\mathbf{B}$ we can as near as possible recover the change in scale that occurs to the neighbourhood as a result of manifold learning.

Figure 5.2 shows the difference between isomorphic and non-isomorphic scaling. The use of non-isomorphic scaling significantly improves the accuracy of the matching between the two sets of points (a matter discussed in more detail in Section 5.4).

5.3.4 Step 4: Calculate Low-dimensional Representation

Once the above transformations have been calculated we have enough information to approximate the low-dimensional position of the new sample. The low-dimensional position, y^*, of x^* is given by

$$y^* = \mathbf{B}(\mathbf{R}(\mathbf{V}x^* + v)) \tag{5.12}$$

where $\mathbf{B}$, $\mathbf{R}$, $\mathbf{V}$ and v are calculated according to the steps above.

A pseudocode implementation of our developed method is shown in Algorithm 1. The simplicity of the developed approach is shown in the simplicity of the

Algorithm 1 Generalized Out-of-Sample Extension

```
Require: $x^* \in \mathbb{R}^p, \mathbf{X} \in \mathbb{R}^p, \mathbf{Y} \in \mathbb{R}^q, k \ll |\mathbf{X}|$
 1: $idx \leftarrow \mathrm{nn}(x^*, \mathbf{X}, \mathbf{Y}, k)$
 2: $\Lambda\mathbf{V} = \mathbf{C_X}\mathbf{V}$
 3: $\mathbf{Z} = \mathbf{X}_{idx}\mathbf{V}_{1\ldots q}$
 4: $(\mathbf{U}\Sigma\mathbf{V}) \leftarrow \mathrm{svd}(\mathbf{Z}^T\mathbf{Y}_{idx})$
 5: $y^* \leftarrow x^*\mathbf{V}_{1\ldots q}$
 6: $y^* \leftarrow y^* + (\bar{\mathbf{Y}}_{idx} - \bar{\mathbf{Z}})$
 7: $y^* \leftarrow y^*\mathbf{T}$
 8: $\mathbf{B} \leftarrow \mathrm{identity}(q, q)$
 9: $\mathrm{diag}(\mathbf{B}) = [\frac{\mathrm{range}(\mathbf{Y}^1_{idx})}{\mathrm{range}(\mathbf{Z}^1_{idx})} \cdots \frac{\mathrm{range}(\mathbf{Y}^q_{idx})}{\mathrm{range}(\mathbf{Z}^q_{idx})}]$
10: $\mathbf{T} \leftarrow \mathbf{U}\mathbf{V}^T$
11: $y^* \leftarrow y^*\mathbf{B}$
12: return $y^*$
```

pseudocode in Algorithm 1. The two main ‘bottle necks’ are the computation of the k-nearest neighbours and the Singular Value Decomposition. However, with efficient implementations of these two steps the run-time of the algorithm is significantly less than that of Bengio or Yang’s solution.

5.4 Results

In this section we discuss the performance of the developed method, GOoSE, against the two main existing approaches: Bengio (Bengio et al., 2003) and Yang (Yang et al., 2010). For the majority of this section we compare GOoSE with Yang’s approach as both provide generalised solutions. However, we do compare with Bengio’s approach when considering the algorithm specific case. We also show how GOoSE can extend other algorithms, in particular our developed approach PLML.

We begin in Section 5.4.1 by defining an embedding error with which we can assess the performance of an out-of-sample extension method. This embedding error is based on measuring the global change between the actual and expected performance of the algorithm. Once an embedding error has been defined we discuss in Section 5.4.2 the effect of different neighbourhood sizes on the performance of GOoSE. Finally, in Section 5.4.3 we compare GOoSE against the two other out-of-sample algorithms by comparing embeddings on a known data set.

5.4.1 Embedding Error

To be able to analyse the performance of out-of-sample extensions we need to first define an embedding error. Given a dataset $\mathbf{D}$ we create a training set, $\mathbf{B}$, and test set, $\mathbf{C}$, such that $\mathbf{B} \cup \mathbf{C} = \mathbf{D}$, $\mathbf{B} \cap \mathbf{C} = \emptyset$ and $|\mathbf{B}| = |\mathbf{D}| - |\mathbf{C}|$. As in (Yang et al., 2010) we can obtain the low-dimensional embedding, $\mathbf{Y}$, by running a manifold learning algorithm on the entire dataset $\mathbf{D}$. We can then express $\mathbf{Y}$ as $\mathbf{Y} = [\mathbf{Y}_{\text{train}}, \mathbf{Y}_{\text{test}}]^T$ where $\mathbf{Y}_{\text{train}}$ and $\mathbf{Y}_{\text{test}}$ are the low-dimensional embeddings of the training and test data. Once $\mathbf{Y}$ is known we can use $\mathbf{B}$ to obtain the training set of the manifold and then use an out-of-sample extension method to estimate the low-dimensional embedding of $\mathbf{C}$. We denote the estimated low-dimensional embedding of the test data $\mathbf{Y}'_{\text{test}}$, we can now define an embedding error based on the root mean square error between the actual and estimated test sets

$$e = \sqrt{\frac{\sum(\mathbf{Y}_{\text{test}} - \mathbf{Y}'_{\text{test}})^2}{n}} \tag{5.13}$$

where n is the number of elements in the test set and both $\mathbf{Y}_{\text{test}}$ and $\mathbf{Y}'_{\text{test}}$ are transformed according to the rotational difference between $\mathbf{Y}_{\text{train}}$ and $\mathbf{B}$ to remove the effect of the manifold learning algorithms mapping the datasets into different low-dimensional spaces[4]. This error measure now provides us with a

[4]This is something that is not considered by Yang et al. in (Yang et al., 2010) but without this step the results obtained are meaningless as the two low-dimensional embeddings are in different co-ordinate spaces

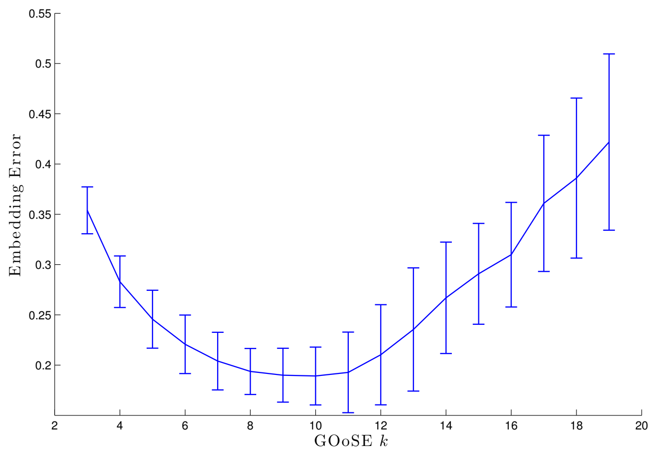


Figure 5.3: The effect of the neighbourhood size parameter k on the embedding error of a dataset with a known low-dimensional manifold.

basis for analysing the performance of an out-of-sample extension method, with a low value of e signifying that the estimated test embedding is closer to the actual test embedding than that of a test embedding with a high value of e.

5.4.2 Parameter Selection

Our algorithm has only one free parameter, the neighborhood size k. To test how this parameter affects the performance we ran a set of experiments on a known manifold with a known low-dimensional embedding. We used the Swiss Roll manifold with 2000 samples and the known low-dimensional embedding. The data was randomly split into training and test sets with each set having a size of 1000. For each permutation of training and test we used the GOoSE algorithm to try and embed the test set into the low-dimensional space with varying parameters of k within the range $[3, 19]$. The embedding error (Eq. 5.13) between the estimated and actual test sets were recorded over 10 runs.

Figure 5.3 shows the results of this test. The graph is shown with associated error bars indicating the standard deviation of the results per value of k. The results show that a minimum is reached around $k = 10 \pm 2$, after this point the RMSE increases along with the standard deviation meaning that results obtained with a larger value of k are more unstable. Figure 5.4 shows the scatter plot of all 190 runs of the experiment. The standard deviations are increased in a low range of k by one or two outliers, the main grouping of points in the range $k = [3, 10]$ are far tighter than those in the range $k = [11, 19]$. This suggests that with the exception of a few outliers the performance of the GOoSE algorithm is more stable at a lower value of k as opposed to a higher value. Although this optimum value of k will change depending on what dataset is used, experiments do support the argument that a local minimum will always exist. Since the GOoSE algorithm is fast to run it is easy to find an optimum value of k by performing a simple parameter search over a range of k values and recording the best performance.

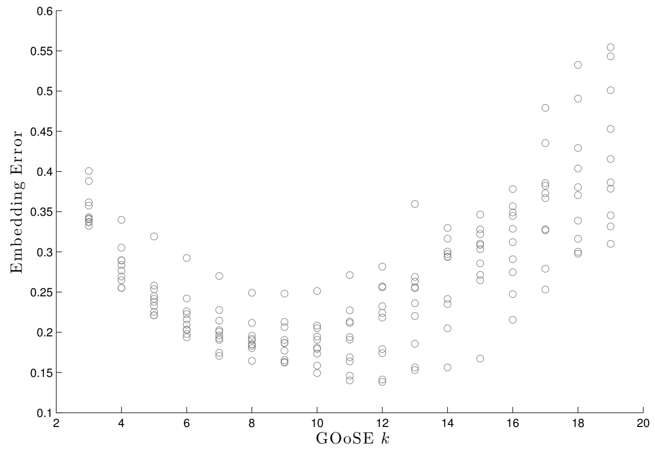


Figure 5.4: The effect of the neighbourhood size parameter k on the embedding error of a dataset with a known low-dimensional manifold.

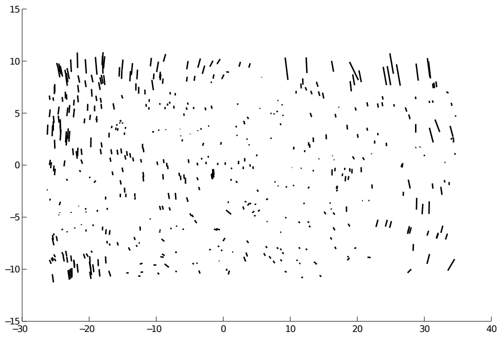
5.4.3 Analysis

In this section we visually compare the performance of the three out-of-sample techniques by visualising the change between the estimated and actual low-dimensional embedding of the test set. This visual comparison is performed using flow diagrams where a line is drawn from the estimated data point to its actual position. The flow diagrams then provide a good way of showing us where the algorithms work well and where they fail.

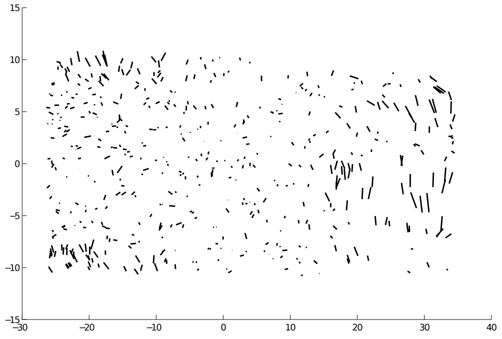
We begin by comparing the performance of all three out-of-sample algorithms when using Isomap. Then we move on to analyse the performance of our developed approach and Yang's approach when using the PLML algorithm. This not only enables us to draw conclusions as to the effectiveness of GOoSE with PLML, but also allows us to see how our approach and Yang's approach work with a global alignment of local models manifold learning algorithm.

Figure 5.5 shows the flow diagrams obtained when running the out-of-sample extension methods on an embedding produced using Isomap. The Isomap parameter was set at $k = 12$, Yang's parameters were set, in accordance to the values suggested in (Yang et al., 2010), to $\gamma = 10^{-4}$ and $\sigma = 10$, while GOoSE's parameter was set to $k = 10$ in accordance to the discussion in Section 5.4.2. A training set of 1500 points was used with a 500 point test set. The size of this split was set so as to enable the training set to have enough data points to fully recover the shape of the manifold. Any size less than this and the training set embedding would differ greatly to the combined training and test embedding.

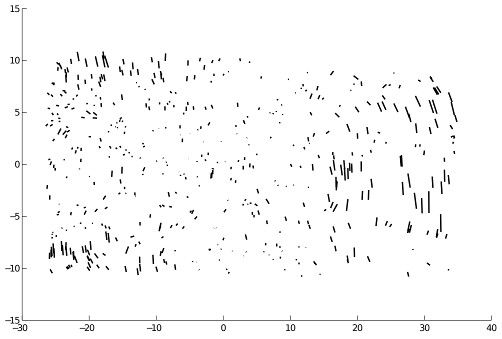
Inspection of Figure 5.5 shows that none of the 3 algorithms manage to perfectly embed the new data points into this manifold. This is not surprising as the manifold is not uniformly sampled and is not 'perfectly' re-constructed in the low-dimensional space. Distortions in the local geometry and global consistency



(a) Bengio



(b) Yang



(c) GOoSE

Figure 5.5: Flow diagrams for the performance of the three out-of-sample techniques on the Swiss Roll dataset reduced using Isomap (with $k = 12$). (b) was obtained with $\gamma = 10^{-4}$ and $\sigma = 10$. (c) was obtained with GOoSE's $k = 10$. The error values for each method are (a) $e_{\text{Bengio}} = 0.4256$, (b) $e_{\text{Yang}} = 0.4210$ and (c) $e_{\text{GOoSE}} = 0.3736$

of the manifold will mean that all three out-of-sample methods will be unable to fully recover the test sample's true positions. This aside, GOoSE manages to out perform both Bengio and Yang's techniques. This is particularly apparent in the central region of the embedding where the flow lines for GOoSE are smaller than that of Bengio and Yang. As well as this the error measure (Eq. 5.13) for the three embeddings are: $e_{\text{Bengio}} = 0.4256$, $e_{\text{Yang}} = 0.4210$ and $e_{\text{GOoSE}} = 0.3736$. Although visually the three embeddings appear quite similar the error measure reveals that GOoSE is able to recover a better approximation of the test set than Bengio and Yang.

The results in Figure 5.5 also reveal some interesting properties of the three out-of-sample algorithms. All of the three approaches struggle in sparsely sampled regions (as displayed by the long lines in Figure 5.5). This is not surprising for GOoSE as it heavily relies on the local geometry of the new sample's neighbourhood to reconstruct the new sample in the low-dimensional space. If this neighbourhood is sparsely sampled then the geometry will be difficult to recover. Having said this, it is still useful to know that GOoSE works well with a global manifold learning method where the local structure of the manifold is not guaranteed to be reconstructed in the low-dimensional space. Even though Isomap is a global technique that introduces local distortions GOoSE is still able to out perform the other two techniques and approximate the positions of the test data points.

Figure 5.6 and Figure 5.7 shows the results of running GOoSE and Yang's method on an embedding learnt using the PLML algorithm. The PLML parameter was set at $k = 9$ as this was recorded as an optimal parameter for this data set in Chapter 4, Yang's parameter values were $\gamma = 10e^{-4}$ and $\sigma = 2$ in accordance to the values suggested in (Yang et al., 2010). GOoSE's parameter was set to $k = 10$. As can be seen from Figure 5.7 GOoSE is a significant

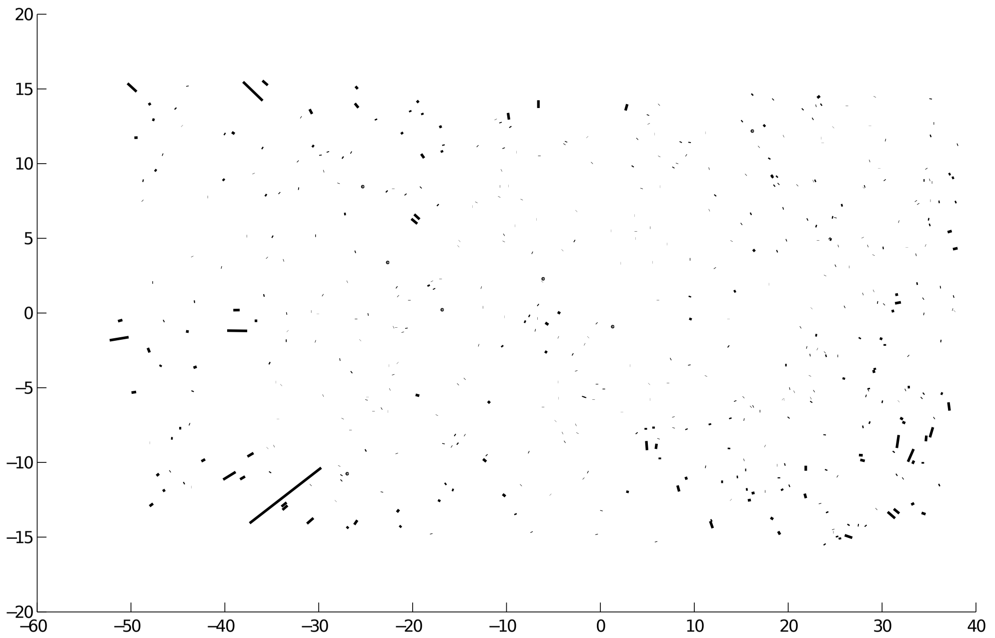


Figure 5.6: Flow diagram for the performance of Yang's out-of-sample technique on the Swiss Roll dataset reduced using PLML (with $k = 9$). The result was obtained with $\gamma = 10^{-4}$ and $\sigma = 2$. The error for this embedding equals $e_{\text{Yang}} = 0.3303$

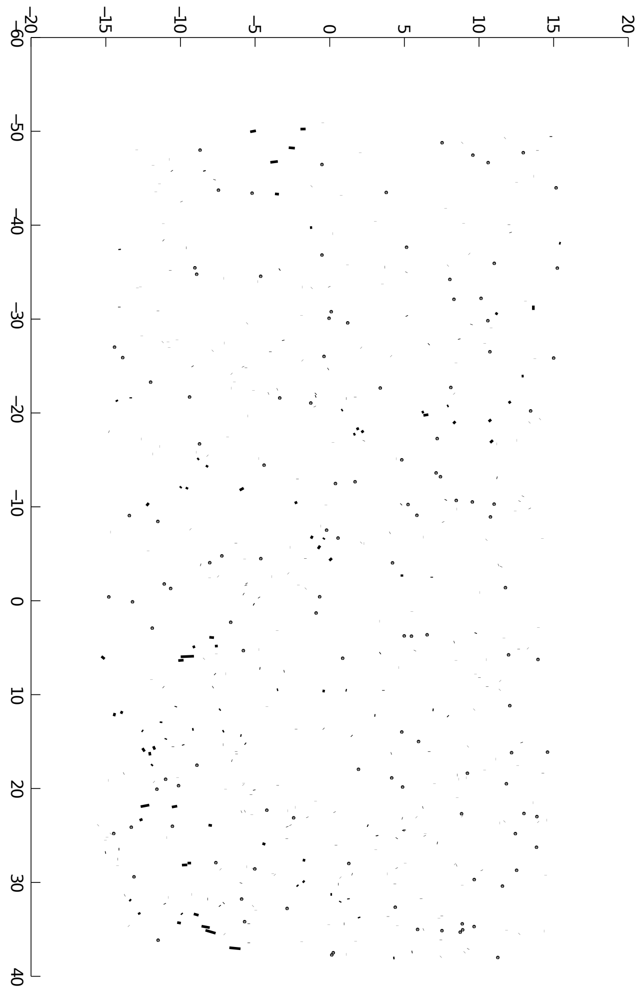


Figure 5.7: Flow diagram for the performance of the GOoSE algorithm on the Swiss Roll dataset reduced using PLML (with $k = 9$). GOoSE's parameter was set to $k = 10$. The error value for this embedding equals $e_{\mathrm{GOoSE}} = 0.0866$.

	Yang (Yang et al., 2010)	GOoSE
Isomap	0.273 ± 0.01	0.138 ± 0.01
LLE	0.014 ± 0.00	0.000 ± 0.00
LTSA	0.022 ± 0.00	0.000 ± 0.00
PCA	0.319 ± 0.03	0.085 ± 0.01
PLML	0.308 ± 0.10	0.085 ± 0.01

Table 5.1: Average embedding error with the Swiss Roll learnt using different manifold learning algorithms. The results were averaged over 10 different folds with a different training and test set used for each fold. Yang's parameters were: $\gamma = 10e^{-4}$, $\sigma = 10$) and GOoSE used $k = 10$.

improvement over Yang's embedding. The change between the actual test embedding and that obtained using GOoSE is much smaller than the difference between the actual embedding and Yang's method. The error values for the two methods are: $e_{\text{Yang}} = 0.3303$ and $e_{\text{GOoSE}} = 0.0866$. GOoSE works well with PLML as they both exploit the local structure of the manifold. At a local scale distortions are kept to a minimum when using PLML as long as the local linearity assumption holds. This assumption is exploited by GOoSE and so when used in conjunction with PLML GOoSE is able to obtain good approximations of the test sets position in the low-dimensional space.

Table 5.1 shows the average runs obtained using a 10-fold cross validation approach. We compare GOoSE with Yang's method using the manifold learning algorithms we compared in Chapter 4. The data was randomly split into 10 folds with 9 being used for training and 1 for testing. This was then repeated until all folds had been used as a test set. The results show that for every manifold learning algorithm GOoSE is able to outperform Yang's method. For LLE and LTSA, GOoSe performs particularly well producing the best results across all manifold learning algorithms. The fact that the pairing of PLML and GOoSE do not achieve this level of performance can be due to the way that PLML models the local structure of the manifold. PLML enforces a hard partitioning of the manifold into local models where as for LLE and LTSA this partitioning is soft. This means that within the hard partitions PLML is able to recover well the local structure of the manifold. So if a new sample's neighbourhood coincides with such a hard partitioning then the GOoSE/PLML combination will work well. However, if the neighbourhood lies across multiple models then the performance will drop. A possible future extension to the PLML algorithm would be to introduce soft partitions. Although this might increase the computational complexity of the algorithm it should improve the performance of GOoSE when used in conjunction with PLML.

Even though the GOoSE and PLML combination does not reach the same performance as GOoSE and LLE or LTSA, it is still a significant improvement over the Yang and PLML combination.

5.5 Conclusions

In this chapter we have presented a new generalised solution to the out-of-sample extension problem in manifold learning. Our developed approach, GOoSE (Generalise Out-of-Sample Extension), works by learning the local geometric change that occurs around an unseen data point as a result of manifold learning. This change is learnt by examining the neighbourhood of the new point in the high and low-dimensional space and finding the similarity transform that matches the two neighbourhoods. By applying this similarity transform to the new data point we are able to get an approximation of its low-dimensional position.

One important consideration is that GOoSE assumes that the novel sample lies within the previously learnt manifold. That is, the novel datapoint does not lie outside of the sampling region of the manifold. If the new sample does lie outside of the manifold then GOoSE may produce sub-standard results. The new sample will be projected onto the subspace spanned by its nearest neighbours on the manifold, but if the sample is far away from its nearest neighbours then its low-dimensional position may not be accurate. Having said this, GOoSE will still be able to approximate a low-dimensional position for this new sample and will not completely fail at producing an embedding.

Our experiments show that the GOoSE algorithm is a significant improvement over the existing approaches to the out-of-sample extension problem. What's more, GOoSE works well with our developed manifold learning algorithm PLML. This means that the two could be used together to form an important tool in many pattern recognition, computer vision and machine learning tasks.

I keep six honest serving-men

(They taught me all I knew);

Their names are What and Why and When

And How and Where and Who

Rudyard Kipling (1865 - 1936)

6 Conclusions

In this chapter we draw conclusions as to the work performed in this thesis. We outline the major contributions and findings contained in this body of work. We also discuss areas for future work.

6.1 Summary

In this thesis we have presented a new algorithm for manifold learning, Piecewise Linear Manifold Learning (PLML), along with a generalised solution to the out-of-sample extension problem in manifold learning, GOoSE. Both algorithms use local geometric properties of the manifold. PLML exploits the connectivity of local PCA models to build a global embedding of the manifold. GOoSE uses the change in local geometry between the high and low-dimensional spaces to embed new data points into a previously learnt embedding. Both algorithms can outperform existing state of the art solutions to manifold learning and out-of-sample extension problems.

At the outset of this thesis we defined the criteria that we wanted our manifold learning algorithm to meet: (1) easy to understand, (2) good performance, (3) scalable and (4) stable under different data conditions and parameter values. These criteria have been met as follows. (1) The PLML algorithm is easy to understand as we can trace its execution on a simple 2-dimensional dataset on paper. PLML does not require the solution of a large sparse eigen-problem or a large dynamic programming problem. Rather the execution steps are easy to follow and understand. (2) The PLML algorithm also yields good results. When compared against existing state of the art manifold learning algorithms PLML is often able to outperform the existing algorithms. We have also shown how PLML can be used to extract meaningful structure from high-dimensional image data. (3) Due to the simplicity and design of the PLML algorithm it is also scalable to large scale applications. This is one of the possible future directions of research. By exploiting the parallelisation possibilities of MST construction and changing the merging strategy we would be able to allow PLML to handle large data sets. (4) We have also shown how PLML is stable under different levels of data sparsity, data noise and parameter values. The PLML algorithm performs in a stable, well defined, manner under high noise and high sparsity.

In the remainder of this section we outline in more detail the conclusions drawn as a result of the development of the PLML and GOoSE algorithms.

6.1.1 Piecewise-Linear Manifold Learning

One of the key strengths of the PLML algorithm is its simplicity and ease of understanding. With most existing manifold learning algorithms it is very difficult to trace the execution of the algorithm as they are often concerned with the minimisation of a cost function obtained through the solution of a sparse eigen-problem. PLML on the other hand builds the global low-dimensional embedding in an iterative manner by merging local models in a topologically pre-defined manner according to the Minimum Spanning Tree of the models. This makes it easier to trace the execution of the PLML algorithm. PLML also does not require the solution to a large dynamic system which simplifies its computational complexity and runtime. If appropriate algorithms are used for the building of the Minimum Spanning Tree (e.g. Parallelised MST (Chon et al., 2001)) and also for the matrix multiplication (e.g. the Strassen algorithm (Strassen, 1969) or Coppersmith-Winograd algorithm (Coppersmith and Winograd, 1990)) then PLML can be used for large datasets which normal manifold learning algorithms with runtime of $O(n^3)$ would not be able to handle.

In this thesis we have also examined the performance of the PLML algorithm under different levels of data sparsity, data noise and parameter selection. Unsurprisingly, PLML performs best under low noise, low sparsity, conditions. However, under increased noise and increased sparsity PLML's performance degrades in a well defined manner (as shown in Table 4.1 and Figure 4.13). We showed that this is not always the case with other manifold learning algorithms. This well defined performance under the presence of noise and sparsity is a useful property as we know that the performance of the PLML algorithm will not suddenly drop out at certain noise or sparsity levels. We have also shown that in the presence of dense data, PLML's parameter value, k, does not affect the quality of the embedding as much as other technique's parameter values. However, with sparse datasets we show that it is worth decreasing the value of k to improve results, and with noisy data it is worth increasing k to counter the effect of noise (See Section 4.3.3.1).

The tests carried out on image datasets reveal that PLML is able to recover meaningful embeddings from high-dimensional image data. Of particular interest is the embedding produced of the Frey faces dataset where the different facial expressions are contained within different areas of the low-dimensional manifold (Figure 4.26). However, results on some image data also reveal how PLML is unable to cope with manifolds with essential loops (e.g. circular, spherical, etc.). This is due to the use of the MST as the skeleton for the topology of the data. The MST will always introduce cuts into the manifold and as such manifolds with essential loops will never be able to be perfectly embedded into the low-dimensional space.

PLML differs from existing global alignment of local models techniques, such as LTSA (Zhang and Zha, 2004), Manifold Charting (Brand, 2003) and Locally Linear Coordination (Roweis et al., 2002), as it uses a hard partitioning of the data to model the manifold at a local scale. It also does not rely on any form of optimisation to perform alignment of these local models. As such it will not get caught in any local minimum.

6.1.2 Generalised Out-of-Sample Extension

The developed solution to the out-of-sample extension problem, GOoSE, utilises the change in local geometry around a new datapoint to calculate its position in the low-dimensional space. This simplicity leads to a fast to run and effective solution to the out-of-sample problem. Our developed method out-performs existing solutions to the out-of-sample extension problem and can be applied to an embedding learnt from any manifold learning algorithm.

Whereas existing solutions to the out-of-sample extension problem (e.g. (Bengio et al., 2003; Yang et al., 2010)) use global properties of the data via kernel functions, GOoSE exploits the local properties of the data. This focus on the local properties of the data makes it a potentially faster solution to the out-of-sample problem as a large kernel problem involving the entire data is avoided. Not only this, but our results show that GOoSE is able to out-perform existing solutions. The central assumption of GOoSE is that if the new data point lies within the previously sampled manifold region, then only the k-nearest neighbours of the new point will be needed to reconstruct it in the low-dimensional space. This is similar to the reconstruction assumption made by LLE (Roweis and Saul, 2000) and our results show that for the out-of-sample extension problem it is a fair assumption to make as our local approach out-performs the global approaches to the out-of-sample extension problem.

The correct choice of parameter, k, for the GOoSE algorithm will be data specific, but experimental results show that a smaller value of k will yield better results. This is due to the fact that at larger values of k the local linearity assumption fails to hold. As such, approximating the manifold as a linear subspace at large values of k will not hold. Since the GOoSE algorithm is fast to run an optimal parameter can be found for each dataset via a simple parameter search.

6.2 Future Work

In this section we outline the possible areas for future research that extend on the work described in this thesis.

- First, a possible extension to the PLML algorithm is to replace the clustering step with a soft partitioning step in a similar fashion to Manifold Charting (Brand, 2003) and Greedy Procrustes (Goldberg and Ritov, 2009). This soft partitioning could lead to improved performance when combined with GOoSE (as described in Chapter 5). However, by introducing soft partitioning the complexity of the merging step would increase as more traversal steps would take place.

- Following on from this, another possible improvement of the PLML algorithm would be to perform the merging step in the low-dimensional rather than high-dimensional space. This would speed up the merging step as the matrix multiplication would be done using matrices of size $q \times q$ rather than $p \times p$. Therefore, for very high-dimensional data where $p \gg q$ this would significantly speed up the merging step. However, work would need to be done on the alignment of the models in the low-dimensional space as there would be no known frame of reference as to the positioning of the models in this space.
- Extending the GOoSE algorithm to handle the pre-image problem (i.e. the reverse of the out-of-sample extension) would open up many application areas. By reversing the GOoSE algorithm to enable it to find the high-dimensional position of known low-dimensional datapoints we would then have a algorithm within which datapoints can be continuously mapped between the two different spaces. As shown in (Jun and Ghosh, 2010) this would subsequently enable us to perform manifold based classification using the GOoSE algorithm.
- From the above extension to the GOoSE algorithm we could use the extended version, alongside the PLML algorithm, to perform classification of texture data. Much work has been performed on texture classification using textons (Zhu et al., 2002). We expect that this idea could be extended using PLML and GOoSE to that of texton manifolds, where the textures are modelled as manifolds rather than clusters. It is expected that by modelling textures as manifolds, classification performance and texture synthesis should be improved. One interesting such application

area would be mammographic risk assessment and texture segmentation (He et al., 2010). By modelling the different mammographic tissue regions as manifolds rather than groups of textons it would be expected that more robust classification results could be obtained. As well as this, the manifold structure could be exploited to help interpolate between textures (Souvenir et al., 2006) and also synthesise textures (Liu et al., 2006a).

- Finally, it will be important to continue to investigate and help solve the open problems in manifold learning outlined in Chapter 2. As previously discussed, these problems will need to be tackled to help make manifold learning accessible and usable to a wider number of application areas. As such, research into the open problems such as large scale learning, sparsity and quality assessment will be essential to aid further work in this field.

6.3 Final Remarks

The developed manifold learning method presented in this thesis, Piecewise-Linear Manifold Learning, presents a novel approach to the manifold learning problem. Based on the idea of global alignment of local models, it is able to recover the manifold structure at both a global and local scale. Our results show that this piecewise-linear modelling can produce significantly improved results over existing approaches to manifold learning especially when the data is densely sampled. It is also far more stable than LTSA, another global alignment of local models technique, under different density levels, noise levels, and parameter values. The developed method is also able to extract meaningful structure from image databases (a matter discussed in detail in Section 4.4).

The developed out-of-sample extension, GOoSE, is a generalised, local, out-of-sample extension method and is able to out-perform Yang's global out-of-sample extension approach (Yang et al., 2010). By concentrating solely on the local geometric change of the data we are able to embed novel data points into previously learnt embeddings with greater accuracy and at less computational cost. The developed method can also be applied to any embedding produced by any manifold learning algorithm.

A
Publications

Publications arising as a result of the work in this thesis are included over the next few pages. A list of these publications is shown below.

- Harry Strange and Reyer Zwiggelaar, Classification Performance related to Intrinsic Dimensionality in Mammographic Image Analysis, Proceedings of the Thirteenth Annual Conference on Medical Image Understanding and Analysis, 2009, pp.219-223

- Harry Strange and Reyer Zwiggelaar, Iterative Hyperplane Merging: A Framework for Manifold Learning. In Frédéric Labrosse, Reyer Zwiggelaar, Yonghuai Liu, and Bernie Tiddeman, editors, Proceedings of the British Machine Vision Conference, pages 18.1-18.11. BMVA Press, September 2010

- Harry Strange and Reyer Zwiggelaar, Parallel Projections for Manifold

Learning. In Proceedings of the Ninth International Conference on Machine Learning and Applications. Washington DC, December 2010. IEEE Press.

- Harry Strange and Reyer Zwiggelaar, A Generalized Solution to the Out-of-Sample Extension Problem in Manifold Learning. In Proceedings of AAAI2011 the Twenty-Fifth Conference on Artificial Intelligence. San Francisco, CA, August 2011.

Classification Performance related to Intrinsic Dimensionality in Mammographic Image Analysis

Harry Strange[a] and Reyer Zwiggelaar[a*]

[a]Department of Computer Science, Aberystwyth University, SY23 3DB, UK

Abstract. In the problem of mammographic image classification one seeks to classify an image, based on certain aspects or features, into a risk assessment class. The use of breast tissue density features provide a good way of classifying mammographic images into BI-RADS risk assessment classes [1]. However, this approach leads to a high-dimensional problem as many features are extracted from each image. These features may be an over representation of the data and it would be expected that the intrinsic dimensionality would be much lower. We aim to find how running a simple classifier in a reduced dimensional space, in particular the apparent intrinsic dimension, affects classification performance. We perform classification of the data using a simple k-nearest neighbor classifier with data pre-processed using two dimensionality reduction techniques, one linear and one non-linear. The optimum result occurs when using dimensionality reduction in the estimated intrinsic dimensionality. This not only shows that optimum performance occurs when classifying in the intrinsic-dimensional space but also that dimensionality reduction can improve the performance of a simple classifier.

1 Introduction

Mammography remains the main tool used for the screening and detection of breast abnormalities and the development of full field digital mammographic imaging systems has led to increased interest in computer aided detection systems [2]. Radiologists are increasingly turning to such Computer Aided Diagnostics (CAD) systems to assist them in the detection and/or evaluation of mammographic abnormalities [3]. As such the reliability and accuracy of such systems is paramount especially as breast cancer constitutes the most common cancer among women in the European Union [4]. Many CAD systems will attempt to detect and classify mammographic abnormalities such as microcalcifications and masses. However there is a strong correlation between breast cancer risk and breast density [5, 6]. Figure 1 shows 4 mammograms covering a range of breast tissue density [1]. Each of these 4 images represents a different BI-RADS class. The American College of Radiology BI-RADS [7] is a widely used risk assessment model. It aims to classify a mammogram into one of four classes according to breast density. The classes can be explained as follows. BI-RADS I: an almost entirely fatty breast, not dense; BI-RADS II some fibroglandular tissue is present; BI-RADS III the breast is heterogeneously dense; BI-RADS IV: the breast is extremely dense. Although BI-RADS is becoming a radiological standard other risk assessment models exist that aim to classify breasts according to different aspects or features present in the mammogram (e.g. Tabár modelling [8]).

For a CAD system to place a mammogram into one of the BI-RADS classes it will need to use some form of classification algorithm. Many algorithms exist for the purpose of classification and generally they work by building a model of the data from "known" examples (i.e. mammograms with known BI-RADS classes). Using this model the classifier will then be able to assign each new mammogram to a BI-RADS class. It is unrealistic to use the raw mammographic

[*]Email: hgs08@aber.ac.uk, rrz@aber.ac.uk

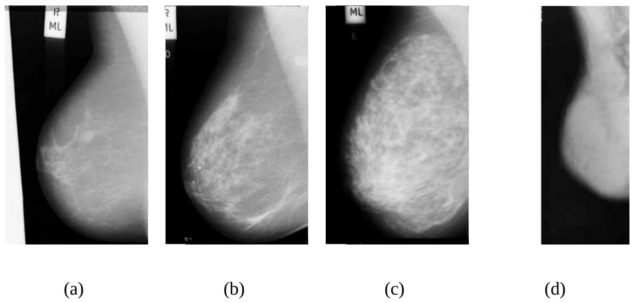


Figure 1. Mammograms showing 4 different breast densities ranging from low density (a) to high density (d).

image as input into a classifier, so features are usually extracted from each image and used as input. In the data from this paper 280 features are extracted from each mammogram (see Section 3). The obvious question to then ask is whether all the extracted features are necessary? Most high-dimensional data will contain redundancy (i.e. dimensions that provide no extra information) which could impair classification performance. Dimensionality reduction is a pre-processing technique that aims to reduce the dimensionality of the data so as to improve classification performance. The question now becomes how many dimensions are needed to best represent the original data? Intrinsic dimensionality estimators aim to find the number of dimensions needed to represent the data without losing important features. In this paper we combine these two elements. We estimate the intrinsic dimensionality of the data and then use dimensionality reduction to show that optimal classification performance occurs when classifying in this intrinsic dimensionality.

We begin by outlining dimensionality reduction and intrinsic dimensionality estimators in Section 2. The data used in this experiment is then discussed in Section 3 before the methodology is outlined in 4. The results are shown in Section 5 before final conclusions and future work are discussed in Section 6.

2 Dimensionality Reduction

Dimensionality reduction is the process of finding from a set of high-dimensional observations a representation of lower dimensionality. This representation will maintain certain aspects, or features, of the original data. Different dimensionality reduction algorithms will retain different features, and this leads to a multi-level taxonomy of techniques. At the highest level techniques can be classified by whether they aim to find a linear subspace within the high-dimensional data, or whether they aim to find a non-linear manifold. We use two dimensionality reduction techniques, one linear (Principal Components Analysis) and one non-linear (Locally Linear Embedding).

2.1 Mathematical Perspective

Given a set of observations $\mathbf{X} = \{x_i\}_{i=1}^{n}$ in an ambient space of dimensionality D (where $x_i \in \mathbb{R}^D$), the aim of dimensionality reduction is to recover the outputs $\mathbf{Y} = \{y_i\}_{i=1}^{n}$ in inherent space d ($d \ll D$ and $y_i \in \mathbb{R}^d$) that best represent the subspace or submanifold contained in the ambient space.

2.2 Principal Components Analysis

Principal Components Analysis (PCA) was first discovered by Pearson in 1901 [9] and was further developed by Hotelling in 1933 [10]. It is perhaps the most widely used dimensionality reduction technique and provides the foundation to many other methods. The principal goal of PCA is to maintain maximal variance between the data points in the low dimensional space and as such it finds the subspace $\mathcal{S}$ within the ambient space that has maximum variance. PCA begins by constructing the zero mean covariance matrix, $\mathbf{C} = cov_{X-\overline{X}}$, of $\mathbf{X}$, before finding the solution to the eigenproblem

$$\mathbf{CW} = \lambda \mathbf{W} \tag{1}$$

The original data, $\mathbf{X}$, is then projected onto the top n eigenvectors of $\mathbf{W}$ to give the low dimensional representation $\mathbf{Y}$.

2.3 Locally Linear Embedding

Locally Linear Embedding (LLE) [11] is one of the more popular non-linear dimensionality reduction techniques. LLE, as the name suggests, aims to preserve the local geometry of the manifold by maintaining local neighborhoods in the high and low dimensional spaces. This is achieved by minimizing the embedding cost function

$$\Psi(\mathbf{Y}) = \sum_{i=1}^{n} |y_i - \sum_{j=1}^{n} \mathbf{W}_{ij} y_j|^2 \tag{2}$$

The weights contained in the matrix $\mathbf{W}$ will have been previously calculated by minimizing a similar reconstruction error based cost function[1]. This can then be minimized by solving an eigenvalue problem whose bottom d eigenvectors

[1] $\varepsilon(\mathbf{W}) = \sum_{i=1}^{n} |x_i - \sum_{j=1}^{n} \mathbf{W}_{ij} x_j|^2$

provide the set of orthogonal coordinates.

2.4 Intrinsic Dimensionality Estimation

An important, but often under used, tool related to dimensionality reduction is the estimation of the intrinsic dimensionality of the data. The intrinsic dimensionality can be defined as the smallest number of independent parameters that is needed to generate the given data set [12]. When using a classifier it is useful to be able to work in the smallest possible dimensionality as high-dimensional problems lead to more redundant data as well as increased computational complexity. If the intrinsic dimensionality can be correctly estimated then the redundant data can be "stripped-away" and the real (intrinsic) data can speak for itself.

Many techniques exist for estimating intrinsic dimensionality (see [13]) and in this paper we use two methods with widely differing approaches. As dimensionality reduction techniques can be broken up into linear and non-linear, so can intrinsic dimensionality estimators. We have chosen one linear and one non-linear. The first method is closely related to PCA and simply uses the Eigenvalues created from Equation 1 to estimate the dimensionality. By calculating the residuals of the Eigenvalues and finding at which point the biggest "jump" from one value to another occurs the intrinsic dimensionality can be estimated. The second is based on the Geodesic Minimum Spanning Tree of the data [14]. This works by creating a sequence of minimal spanning trees using geodesic distances (obtained by the Isomap [15] algorithm) and uses the overall lengths of the minimum spanning trees to estimate the dimensionality of the manifold.

3 Data

The data comes from features extracted from the whole set of 322 mammograms that form the MIAS database [1,16]. The data is based on breast tissue density and consists of 322 samples each with 280 features, 10 from morphological characteristics and the remaining 270 from texture information. A fuzzy C-means approach was used to extract two clusters (relating to fatty and dense tissue) from the mammograms. The morphological features were created using relative area of the fatty and dense clusters as well as the first four histogram moments of these clusters. The texture information was derived from co-occurence matrices [17]. Each of these 322 mammograms have been assigned to a BI-RADS risk assessment class by an expert radiologist [1].

4 Methodology

The first step in this experiment was to obtain classification results using the raw high-dimensional data. A k-fold cross validation technique was employed throughout this experiment. The data was partitioned into two sets: 1 for training the classifier and 1 for testing. The size of each fold was 14 samples. This meant that for each stage of the cross validation experiment 14 of the 322 samples were used for testing the classifier while the remaining 308 were used for training purposes. The average over each of these folds was then used as the high-dimensional result. A simple k-nearest neighbor classifier [18] was used throughout this experiment with the results being averaged over a range of $2 \leq k \leq 30$. The results are averaged so as to try to factor out any effects the classifier parameters might have on the results. We try to solely look at the effects that the dimensionality reduction techniques have. More advanced classifiers could have been used (such as SVM, C4.5 and Bayesian [19]) but the use of these algorithms would have made factoring out parameter effects more difficult.

The outcome of a dimensionality reduction technique is heavily affected by the choice of parameters. So one of the key steps needed when using dimensionality reduction is finding the optimal parameter set. Without this step you run the risk of performing dimensionality reduction at sub-optimal settings, leading to worse classification results. With this in mind a simple parameter search can be used to find the optimal parameters for each technique. For PCA where the only parameter is the target dimensionality the search is straight forward, we simply run the algorithm over a range of dimensions ($1 \leq d \leq 28$). When using LLE the neighborhood size (k) must be specified. So the algorithm was run multiple times over a range of values for k ($2 \leq k \leq 30$) and the optimal value was recorded and used.

Once the optimal parameters have been found the data can then be classified. The results can then be compared against those created in high-dimensional space to see if an improvement occurs. The optimal dimensionality found from the parameter search can also be compared against the estimated intrinsic dimensionality to see if the two do actually coincide.

Ambient Space ($\kappa = 0.50$; $A_c = 56\%$)

	B-I	B-II	B-III	B-IV
B-I	67	30	13	2
B-II	16	62	35	6
B-III	4	11	40	19
B-IV	0	0	7	10
	62%	62%	60%	46%

PCA + k-NN ($\kappa = 0.57$; $A_c = 63\%$)

	B-I	B-II	B-III	B-IV
B-I	63	20	5	2
B-II	16	67	29	4
B-III	8	16	55	14
B-IV	0	0	6	17
	72%	65%	58%	46%

LLE + k-NN ($\kappa = 0.53$; $A_c = 59\%$)

	B-I	B-II	B-III	B-IV
B-I	66	18	2	2
B-II	16	62	36	2
B-III	5	23	50	20
B-IV	0	0	7	13
	76%	60%	53%	35%

Table 1. Confusion Matrices for classification of MIAS database using different dimensionality reduction techniques with optimal parameter sets. The results from classification in high-dimensional ambient space are also shown.

5 Results

The results of the experiments are shown in Table 5 with optimal parameters found to be $PCA(d = 4)$ and $LLE(d = 10, k = 17)$. As well as the confusion matrices the kappa co-efficient and classification accuracy of each experiment is also displayed. The kappa coefficient is a measure of agreement, beyond chance, between the actual results and the predicted results. As can be seen the use of dimensionality reduction improves classification performance over classification in high-dimensional space. PCA gives the best performance with an increase of classification accuracy of 7%. LLE yields an increase of 3%. When examining the kappa co-efficient again PCA yields the biggest improvement signifying that it retains more important aspects of the data between the high and the low-dimensional space. Even though PCA is only a linear technique it still outperforms LLE. One reason for this could be that LLE simply fails to find any meaningful manifold in the high-dimensional space, and so picks up a sub-optimal noisy manifold. Local techniques tend to over fit the manifold and do not necessarily find the global structure of the data.

The graph in Figure 6 shows the classification accuracy of PCA and LLE across a range of dimensions. What is immediately noticeable is the fact that PCA performs best at $d = 4$. After this point there is no noticable change in the classifier's accuracy showing that the data can be well expressed using only 4 dimensions. This correlates with the outcome of the intrinsic dimensionality estimators, both of which estimated the intrinsic dimensionality at 4-dimensions. This gives weight to the fact that optimal classification performance occurs when classifying in the intrinsic-dimensional space. LLE's optimal performance occurs at $d = 10$. The reason for this could be related to the fact that LLE can't find the manifold on which the data lies. The best estimate of the manifold it can find occurs when reducing to 10-dimensions.

6 Conclusions & Future Work

At the beginning of this paper we posed the questions of how well a simple classification algorithm performs in a reduced dimensional space, in particular the apparent intrinsic dimension of the data. From the results shown in Section 5 we can see that when using PCA on feature data extracted from mammographic images the best classification performance occurs when working in the estimated intrinsic dimension. There is also a noticeable increase in the classification accuracy and kappa co-efficient when using either PCA or LLE. This shows that there are benefits to using a classifier in the estimated intrinsic dimension. We intend to extend this work to show how using more advanced classifiers can yield a greater improvement to the classification accuracy.

A further extension to this work would be to look at how different dimensionality reduction techniques affect the classification performance. In this paper we have only focused on two techniques but other methods may be able to pick up more significant aspects (such as topological features) from the data.

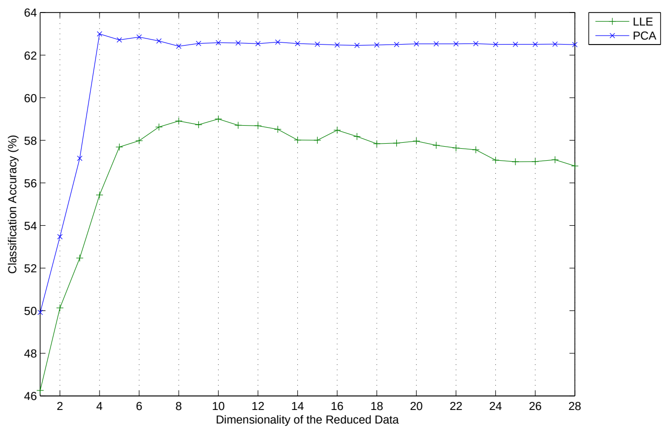


Figure 2. A graph of classification accuracy against dimensionality of reduced data. The optimum of each is PCA($d = 4$) and LLE($d = 10$). The estimated intrinsic dimensionality from both GMST and Eigenvalues was 4. The results from LLE were obtained using $k = 17$.

References

1. A. Oliver, J. Freixenet, R. Marti et al. "A novel breast tissue density classification methodology." *IEEE Transactions on Information Technology in Biomedicine* **12(1)**, pp. 55–65, 2008.
2. C. M. Kuzmiak, G. A. Millnamow, B. Qaqish et al. "Comparison of full-field digital mammography to screen-film mammography with respect to diagnostic accuracy of lesion characterization in breast tissue biopsy specimens." *Academic Radiology* **9**, pp. 1378–1382, 2002.
3. T. W. Freer & M. J. Ulissey. "Screening mammography with computer-aided detection: Prospective study of 12860 patients in a community breast center." *Radiology* **220**, pp. 781–786, 2001.
4. Eurostat. "Health statistics atlas on mortalitiy in the european union." *Office for Official Publications of the European Union* 2002.
5. J. N. Wolfe. "Risk for breast cancer development determined by mammographic parenchymal patterns." *Cancer* **37(5)**, pp. 2486–2492, 1976.
6. N. Boyd, J. Byng, R. Jong et al. "Quantitative classification of mammographic densities and breast cancer risk: Results from the canadian national breast screening study." *Journal of the National Cancer Institute* **(87)**, pp. 670–675, 1995.
7. American College of Radiology. *Illustrated Breast Imaging Reporting and Data System BIRADS*. American College of Radiology, third edition, 1998.
8. L. Tabár, T. Tot & P. B. Dean. *Breast Cancer: The Art And Science of Early Detection with Mammography: Perception, Interpretation, Histopathologic Correlation*. Georg Thieme Verlag, first edition, December 2004.
9. K. Pearson. "On lines and planes of closest fit to systems of points in space." *Philosophical Magazine* **2**, pp. 559–572, 1901.
10. H. Hotelling. "Analysis of a complex of statistical variables into principal components." *Journal of Educational Psychology* **24**, pp. 417–441, 1933.
11. S. T. Roweis & L. K. Saul. "Nonlinear dimensionality reduction by locally linear embedding." *Science* **290**, pp. 2323–2326, 2000.
12. P. J. Verveer & R. P. W. Duin. "An evaluation of intrinsic dimensionality estimators." *IEEE Transactions on Pattern Analysis and Machine Intelligence* **17(1)**, pp. 81–86, 1995.
13. J. A. Lee & M. Verleysen. *Nonlinear Dimensionality Reduction*. Springer, 2007.
14. J. A. Costa & A. O. Hero. "Geodesic entropic graphs for dimension and entropy estimation in manifold learning." *IEEE Transactions on Signal Processing* **52(8)**, pp. 2210–2221, 2004.
15. J. B. Tenenbaum, V. de Silva & J. C. Langford. "A global geometric framework for nonlinear dimensionality reduction." *Science* **290**, pp. 2319–2322, 2000.
16. J. Suckling, P. J, D. Dance et al. "The mammographic images analysis society digital mammogram database." In *Digital Mammography*, pp. 375–378. 1994.
17. R. M. Haralick, K. S. Shanmugan & I. Dunstein. "Textual features for image classification." *IEEE Transactions on Systems, Man and Cybernetics* **SMC-3(6)**, pp. 610–621, 1973.
18. B. V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, 1990.
19. S. Theodorodis & K. Koutroumbas. *Pattern Recognition*. Academic Press, third edition, 2006.

Iterative Hyperplane Merging: A Framework for Manifold Learning

Harry Strange
hgs08@aber.ac.uk
Reyer Zwiggelaar
rrz@aber.ac.uk

Department of Computer Science, Aberystwyth University, Aberystwyth, SY23 3DB, Wales, UK

Abstract

We present a framework for the reduction of dimensionality of a data set via manifold learning. Using the building blocks of local hyperplanes we show how a global manifold can be reconstructed by iteratively merging these hyperplanes. A Minimum Spanning Tree provides the skeleton needed to traverse the manifold so that the local hyperplanes can be used to build a global, locally stable, embedding. We show state of the art results when compared against existing manifold learning approaches using benchmark synthetic data. We also show how our technique can be used on real world image data.

1 Manifold Learning

The area of dimensionality reduction has received much attention over the last few years, thanks in part to the growth in the number of non-linear, or manifold learning, techniques. At its core, any dimensionality reduction algorithm takes a set of high dimensional samples and returns a representation of lower dimensionality that retains certain features found in the high dimensional space. In the simple case a dimensionality reduction algorithm will project the data onto the basis of a global feature, such as the hyperplane of maximum variance (i.e. Principal Components Analysis [9]). More complex algorithms will aim to recover a low-dimensional non-linear manifold embedded in the high dimensional space (e.g. ISOMAP [16]).

It is to this family of manifold learning algorithms that much focus has been given in the computer vision and pattern recognition communities over the past decade. Many problems in these fields require a low dimensional representation to be found as working in higher dimensions can often be problematic [2]. Dimensionality reduction and manifold learning techniques have also been used to reveal patterns in image data [21]. As an example, consider a data set consisting of a sequence of images showing a rotating 3-dimensional object. Across the dataset the object rotates around one of its axes. If each of these images were to be thought of as a point in high-dimensional space (the dimensionality being equal to the number of pixels in the image) then they would lie on a simple circular manifold that is parameterized by the degree of rotation of the object. This means that each image can be discriminated using only 1-dimension - the degree of rotation - as opposed to, in the case of a 128×128px image, $16,384$ dimensions. This reduction of dimensionality overcomes many computational and mathematical problems associated with high-dimensional learning [2, 6].

The simplest form of dimensionality reduction is one in which the non-linearity of the data is ignored and the entire data is projected onto a single linear basis. This approach, known as Principal Components Analysis (PCA), was first developed by Hotelling in 1933 [9] and has been widely used since (e.g [10, 18]). It finds the global hyperplane of maximum variance across the data and projects all points onto the low-dimensional basis vectors of this hyperplane. A covariance matrix is constructed across all the samples and the eigenvectors of this matrix provide the linear basis from which the low-dimensional projection matrix can be formed. If the data set is inherently linear then PCA will work well, but few real world data sets are purely linear and as such PCA will fail to find an optimal embedding of the data. This linear limitation of PCA has lead to much research into non-linear, or manifold learning, techniques (e.g. [14, 16, 22]). All of these techniques work on the assumption that the data lie on a learnable low-dimensional manifold embedded within the high-dimensional input space. For example, ISOMAP [16] constructs a geodesic distance graph across the data to approximate distances across the manifold. The eigenvectors of this geodesic distance matrix form the low-dimensional basis upon which the high-dimensional data can be mapped. Since ISOMAP considers the distance information across the manifold it is seen as a global method for manifold learning (as it maintains a global property - the geodesic interpoint distances). Conversely there exist methods which seek to maintain local properties of the data, such as Locally Linear Embedding [14] which aims to maintain local interpoint relations between the high and low-dimensional spaces. Recently a class of manifold learning algorithm has emerged that seeks to combine the strengths of global and local techniques. This family of global-local techniques aim to preserve both local and global properties of the data. One such example is Local Tangent Space Alignment (LTSA) [22]. LTSA constructs local models around each sample based on its local tangent space. These tangent spaces are then globally aligned via the solution of a minimization problem to produce the low-dimensional embedding of the high-dimensional data. A more recent suite of techniques proposed by Goldberg *et al*. in [7] use PCA and Procrustes analysis to build and align local models. The main algorithm presented, Greedy Procrustes (GP), finds the embeddings of neighborhoods iteratively using PCA and then aligning them to neighboring neighborhoods using Procrustes analysis.

Our technique, Iterative Hyperplane Merging, continues this idea of globally aligning local models using PCA and Procrustes. A clustering algorithm is used to partition the data into local models. PCA is run on these local models to produce local low-dimensional hyperplanes. These local hyperplanes are then iteratively merged to produce a global alignment of the local models. As such Iterative Hyperplane Merging is able to preserve the local properties of the data across a global scale.

2 Iterative Hyperplane Merging

Iterative Hyperplane Merging (IHM) can be intuitively thought of as a form of local PCA where PCA is applied at a local scale to produce low-dimensional local hyperplanes. These local hyperplanes are then globally aligned to produce the final low-dimensional embedding. A clustering algorithm is employed to create the partitions needed to form the local hyperplanes and a Minimum Spanning Tree (MST) is used as the basis for forming the global alignment of the hyperplanes.

The clustering step is used to capture local information and form the local hyperplanes. The lack of overlap between local models is novel when compared against many existing local-global techniques (e.g. [5, 7, 22]) and aims to reduce any within model distortion. This reduction in local distortions leads to a more locally faithful embedding.

To obtain a faithful global embedding we perform a pre-order traversal on the Minimum Spanning Tree (MST) of the inter-hyperplane distance graph. When walking from one node in the MST to another we merge the hyperplanes gradually building a global embedding of the data. The Minimum Spanning Tree has many properties that are well suited to our algorithm. Firstly, as outlined by Robins in [13], the MST provides all the information needed to describe the connectedness of the data. Since topology can be, at least simplistically, thought of as the connectedness of a data the MST provides a good approximation for the topology of a manifold. However, there is insufficient metric information contained within the MST to be used as a geodesic graph for isometric techniques (e.g ISOMAP[16]). Secondly, the non-cyclic nature of the MST ensures that when traversing the tree we won't get caught in any local cycles and every vertex in the tree will be visited in the correct 'topological order'.

2.1 Algorithm

We take as input a high-dimensional set of samples $\mathbf{X} = \{x_i\}_{i=1}^{n} \in \mathbb{R}^q$ sampled from a low-dimensional manifold $\mathscr{M} \in \mathbb{R}^p$ embedded within $\mathbb{R}^q$ (where $p \ll q$). The goal of any manifold learning algorithm is to recover a set of samples $\mathbf{Y} = \{y_i\}_{i=1}^{n} \in \mathbb{R}^p$ from $\mathbf{X}$ that best approximate the p-dimensional manifold $\mathscr{M}$. We assume that at a local scale the manifold $\mathscr{M}$ is homeomorphic to Euclidean space and is a C^∞-manifold (i.e. it is smooth differentiable).

The q-dimensional set of samples $\mathbf{X}$ can be partitioned into k-local hyperplanes by initially using any clustering algorithm to partition the data. Throughout the rest of this paper we will be using either a Gaussian Mixture Model clustering scheme or a constrained k-means clustering algorithm so we will briefly outline the basic methodology of each here[1]. For Gaussian Mixture Modelling we are interested in a particular Gaussian Mixture Model (GMM) where the number of components is equal to n and the parameters are defined as $\Theta = \{\Theta_{i|c}, i = 1, \ldots, n\}$ where $\Theta_{i|c}$ is the mean and covariance matrix of the ith Gaussian density function and $c = 1, \ldots, k$. The output of the likelihood GMM function related to a partition ω_c is a weighted sum of n component densities:

$$p(x \mid \omega_c) = \sum_{i=1}^{n} P(\Theta_{i|c} \mid \omega_c) p(x \mid \omega_c, \Theta_{i|c}) \tag{1}$$

where $P(\Theta_{i|c} \mid \omega_c) = \mu_{ic}$ is the prior probability of the ith component parameter. These mixture parameters are chosen such that $\sum_i \mu_{ic} = 1$. By fixing the means and covariance matrices of each partition we can now assign a sample to partition ω_l if:

$$\omega_l = \arg\max_{y_c} p(x \mid \omega_c) \tag{2}$$

The weights $\mu = \{\mu_{ic}\}$ of the GMM functions $p(x \mid \omega_l), l = 1, \ldots, k$ can be found by solving a constrained minimization function [15]. For constrained k-means we wish to find a set of partitions, $\omega_1, \omega_2, \ldots, \omega_k$, such that the distance between each point, x_i, and its nearest

[1] For a full description of both of these algorithms we refer the reader to [4, 15] for Gaussian Mixture Modelling and [3, 17] for constrained k-means clustering.

partition center, $\bar{\omega}_c$, is minimized:

$$\min_{\omega_1,\ldots,\omega_k} \sum_{i=1}^{n} \min_{h=1,\ldots,k} \left(\frac{1}{2} \parallel x_i - \omega_h \parallel_2^2 \right) \tag{3}$$

with the specific constraint that no cluster, ω_c, is smaller than the minimum cluster size, h, $\mid \omega_c \mid \geq h$.

The clustering step described above produces a partitioning, Ω, such that $\bigcup \Omega = \mathbf{X}$ and for any two distinct partitions, $\omega_i \in \Omega$ and $\omega_j \in \Omega$, $\omega_i \cap \omega_j = \varnothing$. The partitioning should be chosen such that as far as possible $\mid \omega_1 \mid \approxeq \mid \omega_2 \mid \approxeq \ldots \approxeq \mid \omega_k \mid$ although the ability to achieve this is heavily dependant on the choice of clustering algorithm. Since we assume that the manifold $\mathscr{M}$ is locally linear these partitions can be used to find the k-local hyperplanes of the data set. We define a hyperplane as

$$\Pi_i \ = \ \{ \forall x \mathbf{U}\mathbf{U}^T ; \ \bar{\omega}_i \mid x \in \omega_i, \lambda \mathbf{U} = \mathbf{C}\mathbf{U} \} \tag{4}$$

where $\bar{\omega}_i$ is the mean of the samples in ω_i, $\mathbf{C}$ is the covariance matrix of the samples in ω_i and $\mathbf{U}$ is a matrix containing as columns the top p-dimensional eigenvectors sorted according to their associated eigenvalues, λ. The k-hyperplanes are therefore now intrinsically p-dimensional but are still embedded within the q-dimensional space. To globally align these local hyperplanes we first need to find their local connectivity. To do this we construct a Minimum Spanning Tree (MST) [8] across the data using the means of the hyperplanes as vertices for the tree. This ensures that all vertices are connected with minimum cost. The MST is found by firstly connecting all vertices to form a dense graph $G =< V, E >$ where E is the edgelist connecting all vertices, and the vertex list $V = \{\bar{\Pi}_i\}_{i=1}^{k}$ (where $\bar{\Pi}_i$ is the mean of the hyperplane Π_i). The MST, $T =< V, E' >$, is then a subgraph of G with the same vertex set V but a reduced edge set $E' \subset E$ (where E' is the edge set of minimal cost) [2]. T now provides us with an approximation of the topology of the hyperplanes and thus a coarse approximation of the topology of $\mathbf{X}$ (since there is a direct mapping between the connectivity of the hyperplanes and the connectivity of $\mathbf{X}$).

We can now use T to find the global connectivity of the hyperplanes by walking along T merging hyperplanes across each step. To walk along the MST we use a simple pre-order traversal [19] which ensures that parents are visited before children, and siblings are visited in left-to-right order. To describe the process of pre-order traversal we denote the first child of a node v as first$[v]$. next$[v]$ denotes the next sibling of node v, last$[v]$ denotes the last child of node v and size$[v]$ denotes the number of nodes in the subtree of T rooted at v for all $v \in V$. order$[v]$ gives the order in which v is to be visted. So we visit the first node with order$[v] = 1$, then order$[w] = 2$, until we reach order$[z] = k$. Given a random node, r, set at the root node for traversal, a bijection order $\Psi : V \rightarrow \{1, \ldots, k\}$ is a pre-order traversal of T if order$[r]$ = 1 and

- order[first$[v]$] $=$ order$[v] + 1$ *(if v is not a leaf)*
- order[next$[v]$] $=$ order$[v]$+size$[v]$ *(if v is not a last child)*

for all $v \in V$. Given this bijection order Ψ and the direct mapping between hyperplanes and our MST we can say that Π_{Ψ_1} is the first hyperplane to be visited in the pre-order traversal and Π_{Ψ_k} is the kth hyperplane to be visited. We denote $\Pi_{\Psi_a} \rightarrow \Pi_{\Psi_b}$ as the traversal from the

[2]We omit the full algorithmic outline for forming a MST. For a more detailed description we refer to [8]

ath hyperplane to the bth hyperplane. Since we wish to merge hyperplanes when traversing from one to another we define a function

$$f(\Pi_{\Psi_a} \to \Pi_{\Psi_b}) = \mathbf{A}\mathbf{U}_b\mathbf{U}_b^T + (\bar{\Pi}_b - (\bar{\Pi}_b\mathbf{U}_b\mathbf{U}_b^T)) \tag{5}$$

which maps the hyperplane Π_{Ψ_a} onto the hyperplane Π_{Ψ_b}, where $\mathbf{A} = \forall x \in \Pi_a$ and $\lambda_b\mathbf{U}_b = \mathbf{C}_b\mathbf{U}_b$. Since the mapping is cumulative, with a new local hyperplane being added to the global embedding at each iteration, we can define the globally aligned embedding as a matrix augmentation:

$$\mathbf{Y} \leftarrow f(\Pi_{\Psi_{1\dots(k-1)}} \to \Pi_{\Psi_{(k)}}) \leftarrow f(\Pi_{\Psi_{1\dots(k-2)}} \to \Pi_{\Psi_{(k-1)}}) \leftarrow \dots \leftarrow f(\Pi_{\Psi_1} \to \Pi_{\Psi_2}) \tag{6}$$

It is worth noting at this point an optional extra step that can be used to increase the robustness of an embedding. During the walk along T if last$[v]$ is reached then we need to backtrack until we reach v. Backtracking involves visiting and projecting onto hyperplanes we have already visited. Normally each backtracking step is treated as the same as a forwarding step and the above function $f(\Pi_{\Psi_{a'}} \to \Pi_{\Psi_a})$ is used (where $\Pi_{\Psi_{a'}}$ is the image of Π_{Ψ_a} having gone through iterative projections). However, $f(\Pi_{\Psi_{a'}} \to \Pi_{\Psi_a}) \neq f(\Pi_{\Psi_a} \to \Pi_{\Psi_{a'}})$ since the projection matrix that moves us from the linear subspace of $\Pi_{\Psi_{a'}}$ onto Π_{Ψ_a} is orthogonal to $\Pi_{\Psi_{a'}}$. This means that an extra alignment step is needed so that $f(\Pi_{\Psi_{a'}} \to \Pi_{\Psi_a}) \approx f(\Pi_{\Psi_a} \to \Pi_{\Psi_{a'}})$. The alignment step applies simple Procrustes analysis[11] to translate, scale and rotate $\Pi_{\Psi_{a'}}$ to align to Π_{Ψ_a}. Once the translation vector, scale value and rotation matrices have been found we can align the embeddings by adding an extra constraint to the function:

$$f(\Pi_{\Psi_{1,\dots,a'}} \to \Pi_{\Psi_a}) = \mathbf{b}(\mathbf{A}\mathbf{U}_a\mathbf{U}_a^T + (\bar{\Pi}_a - (\bar{\Pi}_a\mathbf{U}_a\mathbf{U}_a^T)))\mathbf{T} + \mathbf{v} \tag{7}$$

where $\mathbf{b}$ is the isomorphic scale value, $\mathbf{T}$ is the rotation matrix and $\mathbf{v}$ is the translation vector.

Once the traversal algorithm outlined above has finished, with or without backtracking, we are able to obtain the final low-dimensional embedding by performing PCA on the matrix $\mathbf{Y}$

$$\Lambda\mathbf{V} = \mathbf{C}_\mathbf{Y}\mathbf{V} \tag{8}$$

where $\mathbf{C}$ is the covariance matrix of $\mathbf{Y}$ and $\mathbf{V}$ is a matrix containing as columns the top p-dimensional eigenvectors sorted according to their associated eigenvalues, Λ.

3 Results

We use synthetic data to analyse the performance of our algorithm on a dataset with a known manifold. A real world image database is then used to show how our algorithm can handle more difficult manifolds. All our experiments are performed using the MATLAB programming environment. For Gaussian Mixture Modelling we use a MATLAB version of Bouman's C implementation [4] which has the added benefit of being unsupervised so will calculate the optimal number of clusters for a given dataset (with the only parameter being an initial guess of the number of clusters needed).

3.1 Synthetic Data

For synthetic data we use the benchmark Swiss Roll data set [16]. It consists of a highly curved 2-dimensional plane wrapped to a Swiss Roll in 3-dimensional space (See Figure 2(a)). Gaussian Mixture Modelling is used to form the partitions needed to create the local hyperplanes and the backtracking step described in Eq. (7) is used. We compare our technique against three widely used algorithms: Principal Components Analysis (PCA), ISOMAP and Local Tangent Space Alignment (LTSA). A visual comparison of how these algorithms perform acroos a range of noise levels is shown in Figure 2. PCA (Fig. 2(e) - Fig. 2(h)) fails to find the underlying manifold at all noise levels since it is simply applying a linear projection to the data. ISOMAP (Fig. 2(i) - Fig. 2(l)) performs better and at all noise levels is able to unwrap the Swiss Roll, although it does introduce unwanted holes in the manifold so the topology, at least at a local scale, is distorted. LTSA performs well when no noise is present (Fig. 2(m)) but fails to find the correct embedding when significant amounts of noise are added (Fig. 2(o) - Fig. 2(p)). IHM (Fig. 2(q) - Fig. 2(t)) consistently performs well managing to find the global shape of the manifold as well as preserving local relations (although the neighborhoods are somewhat squeezed and expanded at the highest noise level).

Three error measures are used to analyse the performance of a manifold learning technique at maintaining both local and global properties of the data. Mean relative rank errors, trustworthiness and continuity [12, 20], are used to measure the local stability as they compare the differences between local neighborhoods in both the high and low dimensional spaces. Intuitively trustworthiness measures the number of samples that exist in a neighborhood in the high-dimensional space but not in the same low-dimensional neighborhood. Continuity measures the number of samples that have entered the low-dimensional neighborhood and do not appear in the same high-dimensional neighborhood. As such they are a good measure of the local connectivity of the data as they measure the amount of change at

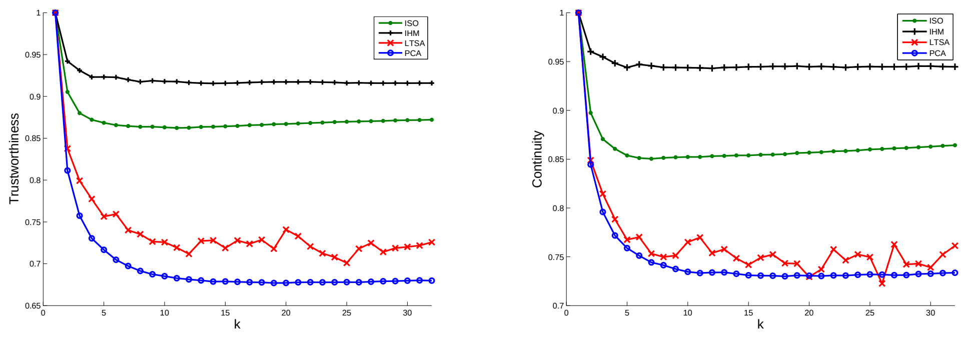


Figure 1: Graph of results for the trustworthiness (left) and continuity (right) of different algorithms when trying to unroll the Swiss Roll dataset with 2000 samples. The neighborhoods for Isomap and LTSA were averaged over the range $k = [2, 32]$. For IHM we use GMM to find optimal cluster sizes with an initial cluster size estimate of 32.

Algorithm	Trustworthiness	Continuity	Mean Square Error
PCA	0.70 ($\pm$ 0.06)	0.75 ($\pm$ 0.05)	0.62 ($\pm$ 0)
ISOMAP	0.87 ($\pm$ 0.02)	0.86 ($\pm$ 0.03)	0.24 ($\pm$ 0.24)
LTSA	0.74 ($\pm$ 0.05)	0.76 ($\pm$ 0.05)	0.23 ($\pm$ 0.36)
IHM	**0.92** ($\pm$ 0.02)	**0.94** ($\pm$ 0.01)	**0.02** ($\pm$ 0.02)

Table 1: Results on Swiss Roll with 2000 samples. The neighborhoods for Isomap and LTSA were averaged over the range $k = [2, 32]$. For IHM we use GMM to find optimal cluster sizes with an initial cluster size estimate of 32.

a local scale. The trustworthiness of an embedding is measured by

$$\mathscr{T} = 1 - \frac{2}{nk(2n-3k-1)} \sum_{i=1}^{n} \sum_{j \in U_k(i)} (r(i,j) - k) \tag{9}$$

Similarly continuity is measured by

$$\mathscr{C} = 1 - \frac{2}{nk(2n-3k-1)} \sum_{i=1}^{n} \sum_{j \in V_k(i)} (\hat{r}(i,j) - k) \tag{10}$$

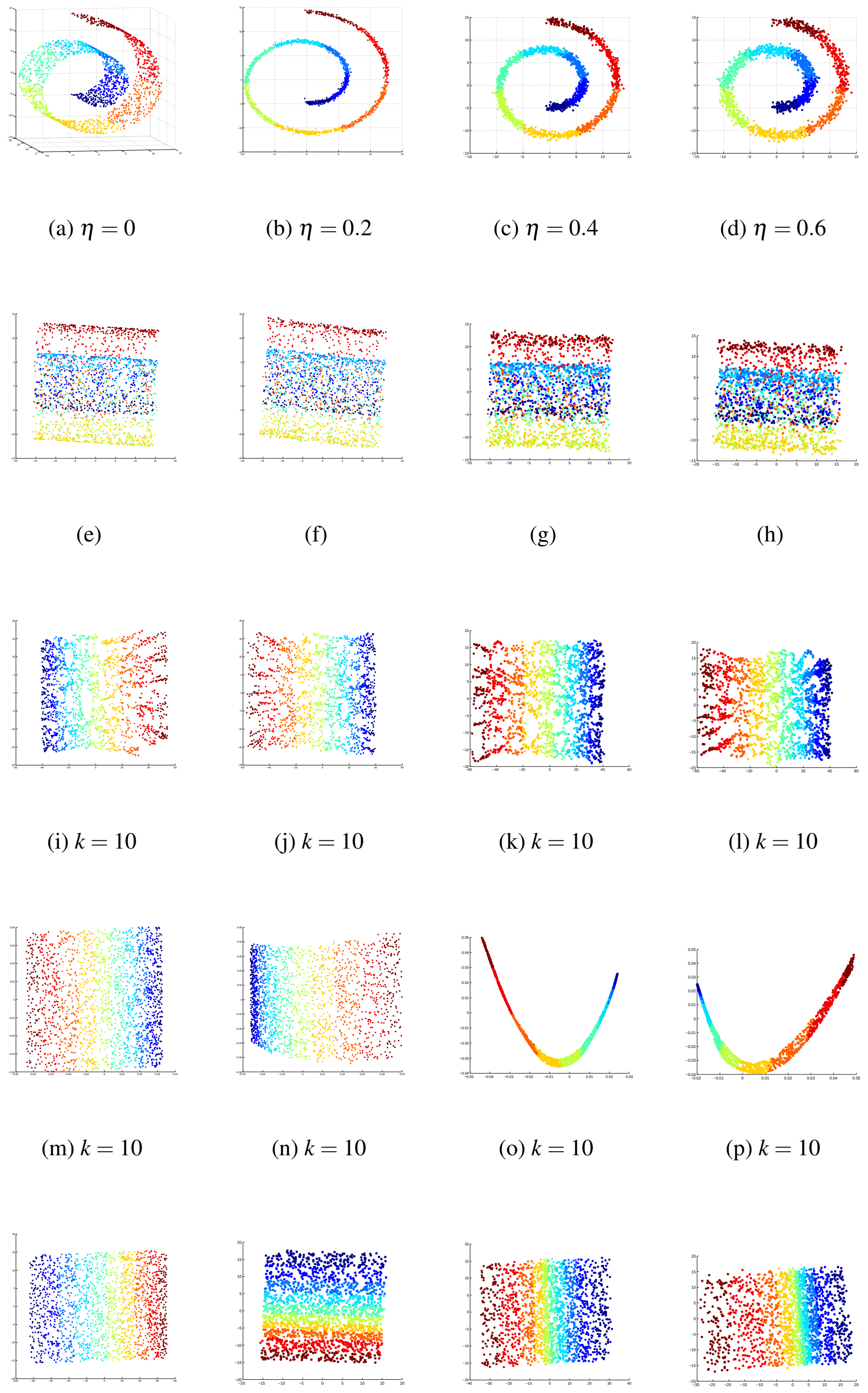
where n is the total number of samples and k is the size of the neighborhoods we wish to measure the trustworthiness and continuity of. $r(i,j)$ is the rank of the data sample $\mathbf{X}_j$ sorted according to the Euclidean distance from sample $\mathbf{X}_i$ in the high-dimensional space, similarly $\hat{r}(i,j)$ is the rank of the data sample $\mathbf{Y}_j$ sorted according to the Euclidean distance from $\mathbf{Y}_i$ in the low-dimensional space. U_k is the set of all samples that in the k-neighborhood of i in the low-dimensional space but not in the high-dimensional space. V_k is the set of samples that are in the k-neighborhood of i in the high-dimensional space but not in the low-dimensional space. Figure 1 shows that over a range of neighborhood sizes IHM is locally stable with the values averaging at $\mathscr{T} = 0.92$ and $\mathscr{C} = 0.94$ (see Table 1). When compared against other techniques IHM, like ISOMAP, is able to consistently maintain local relations with a low variation between embeddings.

Procrustes analysis [11] is used to measure the global stability of an embedding. The embedding, as well as the original unwrapped data, is centered and scaled into a 1×1 square. Procrustes analysis then applies scaling, translation, reflection and rotation in order to minimize the squared distances between an embedding and the original unwrapped data. The lower the sum of squared distances the closer the embedding is to the original data. As shown in Table 1 IHM provides a significant improvement over existing techniques for maintaining global stability. IHM is consistently able to find a globally faithful embedding with low variation between embeddings, as opposed to ISOMAP and LTSA where there is a high variation between embeddings.

3.2 Image Data

To test our algorithm on real world image data we use the Frey faces dataset[3]. The data contains 1965 images of size 20×28 pixels taken from sequential frames of a small video. We use constrained k-means clustering to partition the data as the computational complexity and running time is much reduced when using k-means in high-dimensional space. We also

[3]Frey faces dataset available from `http://cs.nyu.edu/~roweis/data.html`



(a) $\eta = 0$ (b) $\eta = 0.2$ (c) $\eta = 0.4$ (d) $\eta = 0.6$

(e) (f) (g) (h)

(i) $k = 10$ (j) $k = 10$ (k) $k = 10$ (l) $k = 10$

(m) $k = 10$ (n) $k = 10$ (o) $k = 10$ (p) $k = 10$

(q) (r) (s) (t)

Figure 2: (a) 3D Swiss Roll data set with zero added noise, (b-d) the same data with varying uniform noise shown in 2D projection. (e-h) PCA embeddings, (i-l) ISOMAP embeddings with $k = 10$, (m-p) LTSA embeddings with $k = 10$, (q-t) IHM embeddings using GMMs with automatic parameter estimation and an initial cluster size estimate of 32.

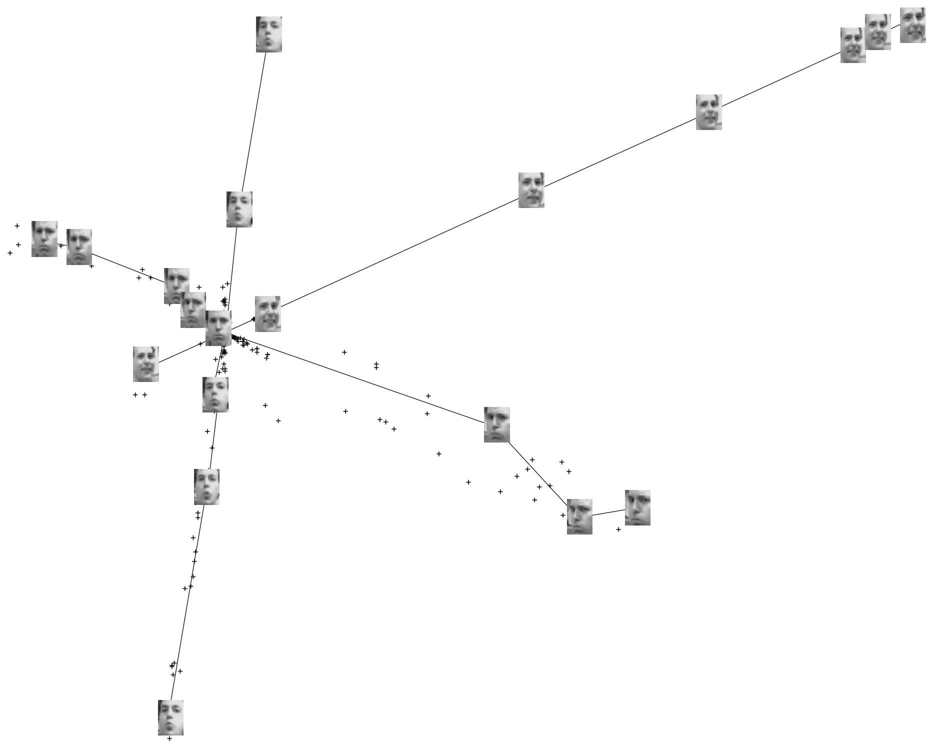


Figure 3: 2-dimensional embedding of the Frey faces dataset found using IHM. The results were obtained using k-means clustering for partitioning, with $k = 64$. The manifold is parameterized by expression with the extremes of the different expressions appearing on the outer edge of the embedding and the central points representing more neutral expression.

omit the backtracking step in Eq. (7) and instead use the forward tracking only method based on Eq. (5). The k-means algorithm was run with $k = 64$ and the minimum cluster size was set to $h = 6$.

The video from which the data is taken shows a face moving through a variety of expressions as well as slight changes in left-right pose. The 2-dimensional embedding found using IHM is shown in Figure 3. The main variation in the embedding is facial expression with the 'extremes' of the expressions (e.g. smiling, frowning) appearing on the edges of the manifold. The points lying in the centre of the manifold are the more neutral, less discriminable, expressions. When compared with other results using this dataset our embedding shows more meaningful structure. The results found in Figure 4 in [7] show that nearby images in the input space match nearby images in the output space, but the output embedding is homogeneous without any apparent of discernable structure. The embeddings found in Figure 3 in [14] reveal slightly more structure in the data but the distribution of expressions is hard to track. Our results show that there is structure in the data with expressions distributed across the manifold and at a local scale images in the local input neighborhood match those in the local output neighborhood.

4 Discussion & Future Work

We have introduced a new unsupervised manifold learning algorithm, Iterative Hyperplane Merging, that is based on the iterative merging of local hyperplanes. By using a clustering algorithm (e.g. Gaussian Mixture Modelling or *k*-means) to partition the data into local hyperplanes, which are then aligned according to a simple walk on the minimum spanning tree of the hyperplanes, we can achieve leading results on benchmark synthetic data. When compared against PCA, ISOMAP and LTSA our algorithm is capable of discovering the shape of the manifold even in the presence of substantial noise. We have shown using various error measures that our algorithm is able to maintain both global and local properties of the data. We have also shown how our algorithm is capable of learning a non-linear face based image manifold parameterized by facial expression.

One limitation of our algorithm is the reliance on a clustering algorithm to partition the samples into local hyperplanes. In high-dimensional spaces this can be both computationally expensive and slow. Also, if the size of the clusters are incorrectly chosen then this can lead to a short-circuited or disconnected manifold [1]. One possible approach to overcome this problem would be to replace the clusters with local tangent spaces. So rather than merging local hyperplanes the local tangent spaces of each sample would be merged. This could provide both an improvement in run-time as well as in embedding accuracy.

References

[1] M. Balasubramanian and E. L. Schwartz. The isomap algorithm and topological stability. *Science*, 295:7a (Techincal Comments), 2002.

[2] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[3] K. P. Bennett, P. S. Bradley, and A. Demiriz. Constrained k-means clustering. Technical Report MSR-TR-2000-65, Microsoft Research, May 2000.

[4] C. A. Bouman. Cluster: An unsupervised algorithm for modeling Gaussian mixtures. Available from http://www.ece.purdue.edu/~bouman, April 1997.

[5] M. Brand. Charting a manifold. In *Advances in Neural Information Processing Systems 15*, pages 961–968. MIT Press, 2003.

[6] D. L. Donoho. High-dimensional data analysis: The curses and blessings of dimensionality. Technical report, Stanford University Department of Statistics, 2000.

[7] Y. Goldberg and Y. Ritov. Local procrustes for manifold embedding: a measure of embedding quality and embedding algorithms. *Machine Learning*, 77(1):1–25, 2009.

[8] R. L. Graham and P. Hell. On the history of the Minimum Spanning Tree Problem. *IEEE Annals of the History of Computing*, 7(1):43–57, 1985.

[9] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[10] R. Huber, H. Ramoser, K. Mayer, H. Penz, and M. Rubik. Classification of coins using an eigenspace approach. *Pattern Recognition Letters*, 26(1):61–75, 2005.

[11] D. G. Kendall. A survery of the statistical theory of shape. *Statistical Science*, 4(2): 87–99, 1989.

[12] J. A. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Springer, 2007.

[13] V. Robins. *Computational Topology at Multiple Resolutions*. PhD thesis, Department of Applied Mathematics at the University of Colorado, Boulder, 2000.

[14] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[15] H. Sahbi. A particular Gaussian Mixture Model for clustering and its application to image retrival. *Soft Computing*, 12(7):667–676, 2008.

[16] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2322, 2000.

[17] S. Theodorodis and K. Koutroumbas. *Pattern Recognition*. Academic Press, 3 edition, 2006.

[18] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pages 586–591, 1991.

[19] G. Valiente. *Algorithms on Trees and Graphs*. Springer-Verlag, Berlin Heidelberg, 2002.

[20] J. Venna and S. Kaski. Local multidimensional scaling. *Neural Networks*, 19:889–899, 2006.

[21] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70:77–90, 2006.

[22] Z. Zhang and H. Zha. Principal manifolds and nonlinear dimension reduction via local tangent space alignment. *SIAM journal on scientific computing*, 26(1):313–338, 2004.

Parallel Projections for Manifold Learning

Harry Strange and Reyer Zwiggelaar
Department of Computer Science
Aberystwyth University
Wales, UK
$\{$*hgs08, rrz*$\}$*@aber.ac.uk*

***Abstract*—Manifold learning is a widely used statistical tool which reduces the dimensionality of a data set while aiming to maintain both local and global properties of the data. We present a novel manifold learning technique which aligns local hyperplanes to build a global representation of the data. A Minimum Spanning Tree provides the skeleton needed to traverse the manifold so that the local hyperplanes can be merged using parallel projections to build a global hyperplane of the data. We show state of the art results when compared against existing manifold learning algorithm on both artificial and real world image data.**

***Keywords*-Manifold Learning; Dimensionality Reduction; Statistical Learning.**

I. Manifold Learning

Many real world problems involve data which lies on, or near, a low-dimensional manifold embedded within a higher-dimensional ambient space [1], [2]. For example, a video sequence of a 3-dimensional object can be represented in a continuous way using far fewer variable than are present in the ambient data. The video sequence can be made up of a set of images each representing a vector in high-dimensional space (where the dimensionality is defined as the number of pixels in each image). Although the ambient dimensionality of this space may be high, the latent dimensionality in reality is much lower as each image will be parameterized by variables such as object orientation, camera position and lighting conditions. Assuming that campera position and lighting are constant then the video sequence could be represented using only 3 variables, i.e. the orientation of the camera. The goal of any manifold learning algorithm is to, as best as possible, recover such low-dimensional manifolds from the high-dimensional data.

The area of manifold learning has become an established field of research with much attention being given to providing algorithms that achieve robust embeddings of highly non-linear data (such as the 3-dimensional object example above). Most, if not all, will have their origins in Principal Components Analysis (PCA) [3] which projects the data onto the hyperplane of maximal variance. As such it is only capable of finding the global subspace of the data and will not find meaningful embeddings for highly non-linear data sets. To overcome this linear constraint many non-linear, or manifold learning, algorithms have been presented that aim to maintain certain properties of the data between the high and low-dimensional spaces. For example, some algorithms will aim to maintain global properties of the data (e.g ISOMAP [1], Kernel PCA [4], and Diffusion Maps [5]), while others will focus on maintaining local properties (e.g. Locally Linear Embedding [2], Laplacian Eigenmaps [6], and Maximum Variance Unfolding [7]). Recently a new class of algorithm has been established that aims to maintain local properties of the data across a global scale. Examples of these global alignment of local models algorithms include Local Tangent Space Alignment [8], Manifold Charting [9], and Local Linear Coordination [10].

In this paper we present a manifold learning technique, Parallel Projections, that fits into this last category of algorithms.

II. Parallel Projections

To obtain a low-dimensional embedding of the high-dimensional data the Parallel Projections algorithm follows four steps. Firstly the local hyperplanes are formed using any clustering method. Then a Minimum Spanning Tree is built between the local hyperplanes so that we can traverse the manifold. The third step involves an iterative walk on the Minimum Spanning Tree where local hyperplanes are merged between node traversals to form a global hyperplane of the manifold. Finally the low-dimensional embedding is found by running PCA on this global hyperplane.

We take as input a high-dimensional set of samples $\mathbf{X} = \{x_i\}_{i=1}^n \in \mathbb{R}^q$ sampled from a low-dimensional manifold $\mathcal{M} \in \mathbb{R}^p$ embedded within $\mathbb{R}^q$ (where $p \ll q$). The goal of any manifold learning algorithm is to recover a set of samples $\mathbf{Y} = \{y_i\}_{i=1}^n \in \mathbb{R}^p$ from $\mathbf{X}$ that best approximate the p-dimensional manifold $\mathcal{M}$. We assume that at a local scale the manifold $\mathcal{M}$ is homeomorphic to Euclidean space and is a C^∞-manifold (i.e. it is smoothly differentiable). A neighborhood size parameter, k, which along with the target dimensionality q are the only free parameters within the developed methodology, is also specified which is used to determine the size of the clusters formed in Step 1.

Step 1: Form Local Hyperplanes. To create the local hyperplanes we firstly calculate the number of clusters $c = \lfloor n/k \rfloor$ and also a minimum cluster size h. This ensures that, as far as possible, all clusters are created with a similar number of samples. In practice the value of h is set to k. We then use a constrained clustering algorithm to partition $\mathbf{X}$ into c clusters $\{C_l\}_{l=1}^{c}$ such that the distance between each point, x_i, and its nearest partition center, $\bar{C}_c$, is minimized:

$$\min_{C_1,\ldots,C_c} \sum_{i=1}^{n} \min_{l=1,\ldots,c} \left(\frac{1}{2} \parallel x_i - \bar{C}_l \parallel^2 \right) \tag{1}$$

with the specific constraint that no cluster, C_l, is smaller than the minimum cluster size, h; $\mid C_l \mid \geq h$. The above approach is similar to that proposed by Bennett [11], where a normal k-means clustering algorithm is run with the specific cluster size constraint added. Similar output can be reached by running a normal clustering algorithm and then finding the small clusters (i.e. $\mid C_l \mid < h$) and joining them to their closest cluster. This process can reduce the number of clusters to fewer than c, but in the remainder of the paper we will still refer to c as the number of clusters. This approach will generally achieve similar results and opens the door for different clustering algorithms to be used.

Once we have partitioned $\mathbf{X}$ into c clusters we are able to form local hyperplanes. A local hyperplane is defined as

$$\Pi_l = \{ \forall x \mathbf{U}\mathbf{U}^T;\ \bar{C}_; \mid x \in C_l,\ \lambda \mathbf{U} = \Sigma \mathbf{U} \} \tag{2}$$

where $\bar{C}_l$ is the mean of the samples in C_l, Σ is the covariance matrix of the samples in C and $\mathbf{U}$ is a matrix containing as columns the top p-dimensional eigenvectors sorted according to their associated eigenvalues, λ. This equates to running Principal Components Analysis on each cluster and then projecting the cluster's samples onto the top $(p+1)$ principal components without changing the cluster's relative position. At this point the data is still embedded within q-dimensional space but is, at a cluster level, p-dimensional. Since the data is locally p-dimensional the next step is to align the local hyperplanes to build a global p-dimensional hyperplane of the data within the q-dimensional space. To do this we firstly form a Minimum Spanning Tree across the hyperplane centres.

Step 2: Build Minimum Spanning Tree. A Minimum Spanning Tree (MST) is a subgraph (which is also a tree) of a connected, undirected graph, with the property that all the vertices are connected together with a minimum sum of edge weights. As pointed out in [12] the MST also has desirable properties that make it ideal as a skeleton of a data set: it tends to avoid shortcuts between branches and it gives a connected graph. It also contains no cycles making it suitable for ordered traversal. The MST is ideal for our application since we wish to find a way of visiting all nodes in a graph at least once and in a well-defined order. We build the MST of the hyperplanes by firstly building an MST of $\mathbf{X}$. If we were to build an MST at hyperplane level the cluster centers would provide too sparse an approximation of the manifold to build a suitably robust graph. As such some of the undesirable properties (like shortcuts) would be likely to return. To build the MST of $\mathbf{X}$ we firstly calculate the $n \times n$ distance matrix $\mathbf{D}$, where $D(i,j) = d(x_i, x_j)$ where $d(x_i, x_j)$ is the Euclidean distance between points x_i and x_j. We can now build the MST, $T = \langle V, E' \rangle$, of $\mathbf{D}$ where T is then a subgraph of $\mathbf{D}$ with the same vertex set V but a reduced edge set $E' \subset E$ (where E' is the edge set of minimal cost)[1]. We can now build the hyperlane level MST, G, given by

$$G(i,j) = \begin{cases} d(\bar{C}_i, \bar{C}_j) & \text{if} \quad \exists e \in E \mid e = \langle v_i, v_j \rangle \\ & \quad\quad v_i \in C_i, v_j \in C_j \\ \infty^{+} & \text{otherwise} \end{cases} \tag{3}$$

where $d(\bar{C}_i, \bar{C}_j)$ is the Euclidean distance between cluster centre $\bar{C}_i$ and cluster center $\bar{C}_j$. We ensure that G is an MST by removing any cycles and ensuring the sum of edge weights is minimal.

Step 3: Forming Global Hyperplane. The main step of the algorithm is to form a global hyperplane across the entire manifold. It is also the most complicated step as it involves local alignments that are different depending on which direction the alignment is being made. The basic idea is to form a global hyperplane by walking along the MST of hyperplanes and merging the hyperplanes as the traversal

[1]We omit the full algorithmic outline for forming an MST. For a more detailed description we refer to [13]

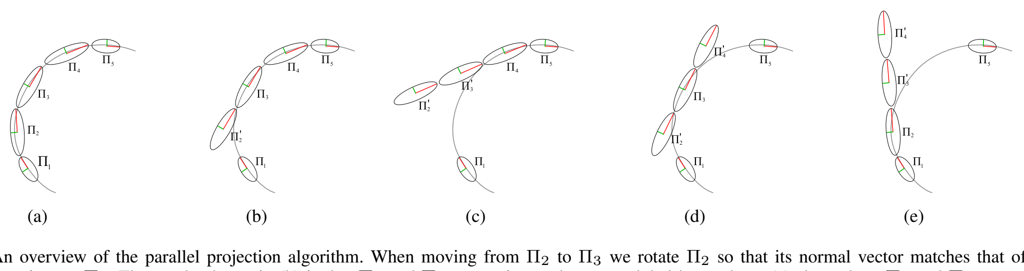


Figure 1. An overview of the parallel projection algorithm. When moving from Π_2 to Π_3 we rotate Π_2 so that its normal vector matches that of Π_3 and then project it onto Π_3. The result, shown in (b) is that Π_2 and Π_3 now exist on the same global hyperplane. (c) shows how Π_2 and Π_3 move onto Π_4. When back aligning, (d), the hyperplane under alignment, Π'_3, is aligned to its original representation Π_3. Similary, when moving back to Π_2, Π'_2 is aligned to Π_2. Notice that throughout this Π_1 and Π_5 remain unchanged as it is not a part of this alignment process.

takes place. Figure 1 shows the principle of this. Starting at hyperplane Π_2 we wish to move to Π_3, so we rotate Π_2 so that it's axis is aligned with Π_3. We then project Π_2 onto Π_3. Since Π_2 and Π_3 are parallel there is no distortion or change in topology within Π_2 which would not be the case were they not parallel. When moving from Π_3 to Π_4 the steps are repeated but the rotation and projection are applied to Π_2 as well as Π_3. If subsequently we want to incoporate Π_1 we need to move back from Π_4 to Π_3 (and subsequently Π_2) since Π_3 (and subsequently Π_2) have already been aligned and is part of the global hyperplane it is a case of translating and aliging Π_3 (and also Π_2) to their original representations. It is worth noting at this point that we build a global hyperplane based on local hyperplanes that are seperate from their original representations. Therefore, we denote Π_i as a hyperplane as originally created and Π'_i as a hyperplane having undergone alignment. So in the last two steps of in Figure 1 we are aligning Π'_3 to Π_3 (and subsequently Π'_2 to Π_2).

To walk along the MST we use a simple pre-order traversal [14] which ensures that parents are visited before children, and siblings are visited in left-to-right order. To describe the process of pre-order traversal we denote the first child of a node v as first$[v]$. next$[v]$ denotes the next sibling of node v, last$[v]$ denotes the last child of node v and size$[v]$ denotes the number of nodes in the subtree of G rooted at v for all $v \in V$. order$[v]$ gives the order in which v is to be visted. So we visit the first node with order$[v] = 1$, then order$[w] = 2$, until we reach order$[z] = c$. Given a random node, r, set as the root node for traversal, a bijection order $\Psi : V \rightarrow \{1, \ldots, c\}$ is a pre-order traversal of G if order$[r] = 1$ and order[first$[v]$] = order$[v] + 1$ *(if v is not a leaf)*; order[next$[v]$] = order$[v]$+size$[v]$ *(if v is not a last child)* for all $v \in V$.

As mentioned above there are two different scenarios that need to be accounted for when moving from one hyperplane to another. If the new hyperplane, Π_i has not been visited before we need to project the previously visited hyperplanes onto this hyperplane (forward projecting). Otherwise, if the hyperplane has been visited before then we need to align Π'_i to Π_i (back aligning). We deal with each of these cases seperately below.

3.1: Forward Projecting. The algorithm for forward projection is based on the need to align the axis of the previously visited hyperplane to the new hyperplane. We denote the previous hyperplane as Π_{prev} and the new hyperplane as Π_{new}. We wish to make Π_{prev} parallel to Π_{new} and to do this we align the axis of Π_{prev} to the axis Π_{new} by finding the rotation matrix needed to rotate the normal vector of Π_{prev} to match the normal vector of Π_{new}. To ensure that no other rotation takes place apart from that defined by the axis of the normal vector we build axis matrices for each hyperplane. For Π_{prev}, with basis vectors $\{u_1, u_2, \ldots, u_p\}$, this is defined as

$$\mathbf{M}^a_{\text{prev}} = \left[\begin{array}{cccc} u_1, & 2u_2, & \ldots, & (d+1)u_{d+1} \end{array} \right] \quad (4)$$

$$\mathbf{M}^b_{\text{prev}} = \left[\begin{array}{cc} -1.5u_1, -2.5u_2, \ldots, & -(d+1.5)u_{d+1} \end{array} \right] \quad (5)$$

$$\mathbf{M}_{\text{prev}} = \left[\mathbf{M}^a_{\text{prev}} \mathbf{M}^b_{\text{prev}}\right] \quad (6)$$

The axis matrix goes up to the $(d+1)^{th}$ principal component since this is the axis that is analogous to the normal vector to the hyperplane. The axis matrix for Π_{new} can be similarly built with the basis vectors of Π_{prev} projected onto the basis vectors of Π_{new} and the normal vector of Π_{new}. This ensures that rotation only occurs in the direction of the normal vector and that no 'skew' is introduced by the rotation. By changing the length of each axis and mirroring the principal axes in the negative direction we can pose the alignment of the two principal axis matrices as a registration problem. Since both axis matrices are centered at the origin the problem is defined as finding the rotation matrix $\mathbf{R}$ so that $\|\mathbf{M}_{\text{new}} - \mathbf{M}_{\text{prev}}\mathbf{R}\|$ is minimized (where $\| \cdot \|$ is the Frobenius norm). The matrix $\mathbf{R}$ is orthonormal and gives the rotation needed to align the axis matrix $\mathbf{M}_{\text{prev}}$ to $\mathbf{M}_{\text{new}}$. Wang et al. [15] have shown that the optimal solution is given by the Singular Value Decomposition (SVD) of $\mathbf{M}^T_{\text{prev}}\mathbf{M}_{\text{new}}$. That is, if the SVD of $\mathbf{M}^T_{\text{prev}}\mathbf{M}_{\text{new}}$ is $\mathbf{U}\Sigma\mathbf{V}^T$, then $\mathbf{R} = \mathbf{U}\mathbf{V}^T$. To apply rotation we simply perform the matrix multiplication $\Pi'_{\text{prev}} = \Pi_{\text{prev}}\mathbf{R}$. Now that the two hyperplanes are parallel we need to update the principal components (axis) of Π_{prev}, that is $\mathbf{U}_{\text{prev}} = \mathbf{U}_{\text{prev}}\mathbf{R}$, and then project Π'_{prev} onto Π_{new}. This projection is given by

$$\Pi'_{\text{prev}} = \Pi'_{\text{prev}}\mathbf{U}_{\Pi_{\text{new}}}\mathbf{U}^T_{\Pi_{\text{new}}} + (\bar{\Pi}_{\text{new}} - (\bar{\Pi}_{\text{new}}\mathbf{U}_{\Pi_{\text{new}}}\mathbf{U}^T_{\Pi_{\text{new}}})) \quad (7)$$

where $\mathbf{U}_{\Pi_{\text{new}}}$ is a matrix containing the top p prinicpal components of Π_{new}. This rotation and projection should be performed using the same rotation matrix for all previously visited hyperplanes, but for simplicity purposes we only show notation for the previously visited hyperplane.

3.2: Back Aligning. If the hyperplane we wish to move to has already been visited then we do not need to worry about the parallel projection process described above. Rather we simply wish to align the hyperplane to its original representation. If we wish to align Π'_i to Π_i then we need to find the translation vector that moves the centroid of Π'_i onto the centroid of Π_i and also the rotation matrix that correctly aligns the samples in Π'_i to Π_i. This step can easily be achieved by running Procrustes analysis on the two sets of samples within the two hyperplanes. Since the number of samples within the hyperplanes will not have changed and the forward projection step described above does not distort the samples within the hyperplane we can use Procrustes analysis [16] to find the translation vector v and the rotation matrix $\mathbf{R}$ to match Π'_i to Π_i [2]. Therefore when back aligning,

[2]We ommit a full discussion of Procrustes here for brevity. A detailed explanation of Procrustes can be found in [17]

$\Pi'_i = \Pi'_i\mathbf{R} + v$. As with the forward projection step the translation and rotation should be applied to all previously visited hyperplanes not just the previously visited one.

4: Find Low-dimensional Embedding. Once all nodes have been visited in the traversal described above, that is all nodes have been visited and in the correct order given by Ψ, we will have obtained a globally aligned p-dimensional representation of the manifold embedded within q-dimensional space. This representation is contained in the hyperplanes $\{\Pi'_l\}_{l=1}^{c}$. We set the matrix $\mathbf{Q} = \{\Pi'_l\}_{l=1}^{c}$ to contain all the samples of the globally aligned representation. To find the low-dimensional embedding we run PCA on $\mathbf{Q}$

$$\Lambda\mathbf{V} = \mathbf{C_Q}\mathbf{V} \tag{8}$$

where $\mathbf{C_Q}$ is the covariance matrix of $\mathbf{Q}$ and $\mathbf{V}$ is a matrix containing as columns the top q-dimensional eigenvectors sorted according to their associated eigenvalues, Λ. The low-dimensional representation $\mathbf{Y}$ is then given by $\mathbf{Y} = \mathbf{QV}$.

III. Results and Discussion

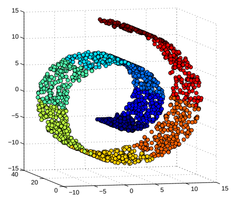
To test the Parallel Projection algorithm we apply it to two known toy data sets and one real world image data set. We compare its performance at learning the toy data sets against nine other leading manifold learning algorithms. Algorithms were chosen from each class of algorithm (as discussed earlier in the paper) so as to obtain a fair as possible comparison. All experiments were performed in the MATLAB programming environment.

A. Toy Data

We test our algorithm on two benchmark toy datasets: the Swiss Roll [1] and a 3-dimensional Möbius strip. Both datasets exhibit interesting manifold structure, the Swiss Roll is a highly curved 2-dimensional plane embedded within 3-dimensional space. As such a simple 2-dimensional projection of the data will not uncover the true structure of the manifold. The Möbius strip can be trivially thought of as a rectangle with one end twisted by $180°$ and then joined to the other end. One interesting property of the Möbius strip is that it is an example of a one sided surface. This makes finding a meaningful 2-dimensional embedding difficult.

Trustworthiness and continuity [19], are used to measure the local stability as they compare the differences between local neighborhoods in both the high and low dimensional spaces. Intuitively trustworthiness measures the number of samples that exist in a neighborhood in the high-dimensional space but not in the same low-dimensional neighborhood. Continuity measures the number of samples that have entered the low-dimensional neighborhood and do not appear in the same high-dimensional neighborhood. As such they are a good measure of the local connectivity of the data as they measure the amount of change at a local scale[3].

[3]The full description of trustworhiness and continuity can be found in [19]


(a)

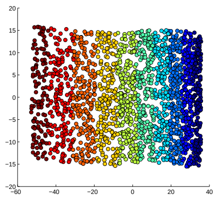

(b)

Figure 2. Results of running Parallel Projections on the Swiss Roll dataset with $k = 10$. Both the local and global properties of the data are correctly kept and the manifold is nicely unfolded with no distortions.

The results of the Parallel Projections algorithm on the Swiss Roll dataset is shown in Figure 2. As can be seen in Figure 2 Parallel Projections find a meaningul and faithful embedding of the Swiss Roll data. Locally neighborhood relations are maintained and globally the shape of the manifold is true to the true 'unwrapped' Swiss Roll (a rectangle with correct class distribution). One of the benefits of Parallel Projections is that it does not normalize the final embedding which can often be problematic [20]. The Möbius strip indicated how the Parallel Projections algorithm works with cyclic manifolds. Due to the use of an MST Parallel Projections will cut the manifold which can be seen as a drawback, but also can be shown to be a benefit as it enables us to visualise the Möbius strip on a 2-dimensional viewing plane[4].

Table I shows the quantitative comparison of Parallel Projections with nine other leading algorithms. We ran the algorithms on the Swiss Roll and Möbius strip datasets over a range of parameters. The trustworthiness and continuity results were obtained by averaging the results over the neighborhood size range of $m = [8, 10, 12, 14, 16]$. The results show that Parallel Projections performs well on both datasets, significantly outperforming other leading algorithms such as ISOMAP and Local Tangent Space Alignment. The global stability of Parallel Projections can be seen in Figure 2 where the global shape of the manifold is maintained with no normalized. In Table I we can see that at a local scale Parallel Projections works well too. This is because locally the data should not change. The parallel constraint on the projection ensures that no distortion or topology change is introduced at a local scale into the final embedding. This makes Parallel Projections a powerful tool for visualisation where it is important to maintain local properties of the data.

B. Image Data

The Frey Faces dataset[5] is used to test our algorithm on real world image data. The data set consists of 1965

[4]Figure not included due to space restrictions

[5]Frey faces dataset available from http://cs.nyu.edu/~roweis/data.html

images extracted from a video sequence of a face undergoing different expressions with slight pose variation. Each image is 20×28 pixels so the ambient dimensionality of the data is 560. Figure 3 shows the results of running the Parallel Projections algorithm on the Frey Faces data set. The embedding is shown on the left of Figure 3 by means of a grid of thumbnails. The embedding space found by Parallel Projections was divided into a grid of 30-by-30 cells. The thumbnail image corresponds to the average face of all samples lying within the given cell. This technique is a useful way of visualising the final embedding. We also manually segmented the gridded data into 'meta' cells based on visual perception of regions within the embedding. These segmented regions were chosen to group together, as far as possible, similar facial expressions.

As can be seen the 2-dimensional embedding seperates out the different facial expressions (as shown in the top right of Figure 3). The manual segmentation shows that different parts of the low-dimensional embedding correspond to different facial expressions with the central point of the entire embedding corresponding to a neutral - expressionless - face. When compared with other results using this dataset our embedding shows meaningful structure. For example, the embeddings found in Figure 3 in [2] reveal some structure in the data but the distribution of expressions is hard to track and nearby points in the output space do not match nearby points in the input space. Our results show that there is structure in the data with expressions distributed across the manifold and at a local scale images in the local input neighborhood match those in the local output neighborhood.

IV. Conclusions & Future Work

We have presented a new manifold learning algorithm, Parallel Projections, which can successfully learn highly non-linear manifolds such as the Swiss Roll and Frey Faces dataset. We tested our algorithm on both these data sets along with the Möbius strip data and showed state of the art results, particularly for the two toy datasets where Parallel Projections outperformed nine other leading manifold learning algorithms. The key strength of the algorithm is that it works well at preserving local structure over a global scale. As long as the data is well sampled and the Minimum Spanning Tree is correctly formed then the Parallel Projections algorithm will work well. If the data is too sparse then the clusters may be incorrectly formed or the Minimum Spanning Tree may create shortcuts across the manifold. Also, if the samples in the cluster are too sparse then it will become difficult to correctly form the principal axis of that cluster leading to inconsistencies in the global alignment step. These aspects will be further investigated. In the future we intend to test Parallel Projections with different clustering algorithms (such as Fuzzy C-means or Gaussian Mixture Modelling). We also intend to try to factor out the clustering step and see if hyperplanes could be formed from local neighborhoods around a sample.

References

[1] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2322, 2000.

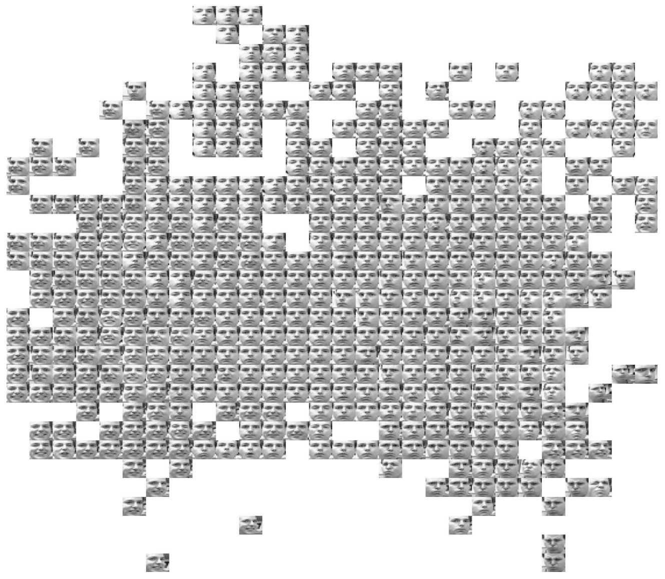
[2] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[3] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, pp. 417–441, 1933.

[4] B. Scholkopf, E. Smola, L. Bottou, C. Burges, H. Bultho, K. Gegenfurtner, and P. H. Ner, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

	Swiss Roll		Möbius Strip	
	Trustworthiness	Continuity	Trustworthiness	Continuity
Diffusion Maps[5]	$0.744(\pm 0.11)$	$0.748(\pm 0.11)$	$0.868(\pm 0.07)$	$0.866(\pm 0.07)$
ISOMAP[1]	$0.939(\pm 0.07)$	$0.940(\pm 0.06)$	$0.848(\pm 0.08)$	$0.845(\pm 0.09)$
Kernel PCA[4]	$0.625(\pm 0.18)$	$0.627(\pm 0.18)$	$0.769(\pm 0.10)$	$0.768(\pm 0.10)$
Laplacian Eigenmaps[6]	$0.811(\pm 0.07)$	$0.814(\pm 0.06)$	$0.757(\pm 0.10)$	$0.748(\pm 0.11)$
Local Linear Coordination[18]	$0.654(\pm 0.18)$	$0.656(\pm 0.18)$	$0.677(\pm 0.20)$	$0.670(\pm 0.20)$
Locally Linear Embedding[2]	$0.783(\pm 0.12)$	$0.784(\pm 0.12)$	$0.866(\pm 0.11)$	$0.865(\pm 0.11)$
Local Tangent Space Alignment[8]	$0.856(\pm 0.06)$	$0.858(\pm 0.06)$	$0.898(\pm 0.05)$	$0.897(\pm 0.06)$
Manifold Charting[9]	$0.747(\pm 0.12)$	$0.749(\pm 0.12)$	$0.912(\pm 0.05)$	$0.912(\pm 0.05)$
Maximum Variance Unfolding[7]	$0.822(\pm 0.17)$	$0.823(\pm 0.17)$	$0.899(\pm 0.08)$	$0.898(\pm 0.08)$
Parallel Projections	$\mathbf{0.982}(\pm\mathbf{0.03})$	$\mathbf{0.983}(\pm\mathbf{0.03})$	$\mathbf{0.992}(\pm\mathbf{0.01})$	$\mathbf{0.992}(\pm\mathbf{0.01})$

Table I
THE RESULTS OF RUNNING VARIOUS MANIFOLD LEARNING ALGORITHMS ON THE TWO TOY DATASETS. THE BEST RESULTS ARE DISPLAYED IN BOLD TYPE. FOR THOSE TECHNIQUES THAT TAKE A NEIGHBORHOOD SIZE AS A PARAMETER (ISOMAP, LAPLACIAN EIGENMAPS, LOCAL LINEAR COORDINATION, LOCALLY LINEAR EMBEDDINGS, LOCAL TANGENT SPACE ALIGNMENT, MAXIMUM VARIANCE UNFOLDING AND PARALLEL PROJECTIONS) THE VALUE OF k WAS USED OVER THE RANGE $[8, 10, 12, 14, 16]$ TO OBTAIN AN AVERAGE RESULT. FOR MANIFOLD CHARTING WE USED 10 FACTOR ANALYSERS. THE PARAMETERS FOR DIFFUSION MAPS, t AND σ, WERE BOTH SET TO 1. THE GAUSSIAN KERNEL WAS USED FOR KERNEL PCA.



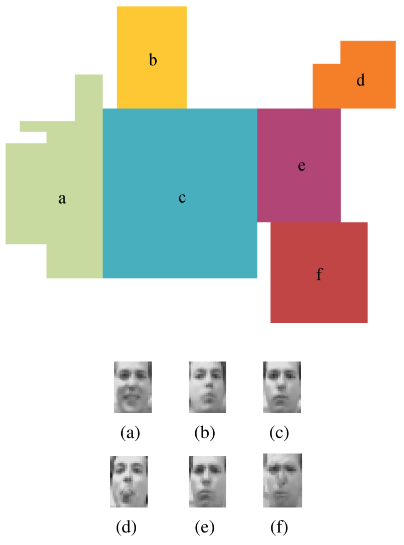


Figure 3. Results of the Frey Faces dataset using the Parallel Projections algorithm with $k = 10$. The left shows the gridded version of the data. Right hand side shows a manual segmentation of the gridded data driven by perception. (a-f) are the median faces from each of the manually segmented regions. The facial expressions are, (a) happy, (b) pouting/looking up, (c) neutral, (d) sticking tongue out, (e) unhappy, (f) frowning. Most of the expressions have paths back to the central neutral expression.

[5] S. Lafon and A. B. Lee, “Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 9, pp. 1393–1403, 2006.

[6] M. Belkin and P. Niyogi, “Laplacian eigenmaps and spectral techniques for embedding and clustering,” in *Advances in Neural Information Processing Systems 14*. MIT Press, 2002, pp. 585–591.

[7] K. Q. Weinberger and L. K. Saul, “An introduction to nonlinear dimensionality reduction by maximum variance unfolding,” in *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006.

[8] Z. Zhang and H. Zha, “Principal manifolds and nonlinear dimension reduction via local tangent space alignment,” *SIAM Journal on Scientific Computing*, vol. 26, no. 1, pp. 313–338, 2004.

[9] M. Brand, “Charting a manifold,” in *Advances in Neural Information Processing Systems 15*. MIT Press, 2003, pp. 961–968.

[10] S. Roweis, L. K. Saul, and G. E. Hinton, “Global coordination of local linear models,” in *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[11] K. P. Bennett, P. S. Bradley, and A. Demiriz, “Constrained k-means clustering,” Microsoft Research, Tech. Rep. MSR-TR-2000-65, May 2000.

[12] M. . Carreira-Perpin and R. S. Zemel, “Proximity graphs for clustering and manifold learning,” in *Advances in Neural Information Processing Systems*. MIT Press, 2005, pp. 225–232.

[13] R. L. Graham and P. Hell, “On the history of the minimum spanning tree problem,” *IEEE Annals of the History of Computing*, vol. 7, no. 1, pp. 43–57, 1985.

[14] G. Valiente, *Algorithms on Trees and Graphs*. Berlin Heidelberg: Springer-Verlag, 2002.

[15] C. Wang and S. Mahadevan, “Manifold alignment using Procrustes analysis,” in *Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland*, 2008.

[16] J. C. Gower, “Generalized procrustes analysis,” *Psychometrika*, vol. 40, no. 1, pp. 33–51, 1975.

[17] T. F. Cox and M. A. A. Cox, *Multidimensional Scaling*. Chapman and Hall, 2001.

[18] J. Verbeek, “Learning non-linear image manifolds by global alignment of local linear models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1236–1250, 2006.

[19] J. Venna and S. Kaski, “Local multidimensional scaling,” *Neural Networks*, vol. 19, pp. 889–899, 2006.

[20] Y. Goldberg, A. Zakai, D. Kushnir, and Y. Ritov, “Manifold learning: The price of normalization,” *Journal of Machine Learning Research*, vol. 9, no. 1909-1939, 2008.

A Generalised Solution to the Out-of-Sample Extension Problem in Manifold Learning

Paper ID: 293

Abstract

Manifold learning is a powerful tool for reducing the dimensionality of a dataset by finding a low-dimensional embedding that retains important geometric and topological features. In many applications it is desirable to add new samples to a previously learnt embedding, this process of adding new samples is known as the out-of-sample extension problem. Since many manifold learning algorithms do not naturally allow for new samples to be added we present an easy to implement generalized solution to the problem that can be used with any existing manifold learning algorithm. Our algorithm is based on simple geometric intuition about the local structure of a manifold and our results show that it can be effectively used to add new samples to a previously learnt embedding. We test our algorithm on both artificial and real world image data and show that our method significantly out performs existing out-of-sample extension strategies.

Introduction

Manifold learning is a widely researched statistical tool used to reduce the dimensionality of a dataset by projecting the high-dimensional data onto a representative low-dimensional manifold. At its simplest form this low-dimensional manifold can be the hyperplane of maximum variance resulting in the data being projected onto a linear basis (Hotelling 1933). More recent techniques aim to find a non-linear manifold upon which the data can be projected (Tenenbaum, de Silva, and Langford 2000; Roweis and Saul 2000). Non-linear techniques are able to discover more complex manifolds than their linear counterparts and so pave the way for manifold learning to be used as a powerful statsticial tool in image processing (Verbeek 2006), data mining (Patwari, III, and Pacholski 2005) and classification (Strange and Zwiggelaar 2009).

One of the open questions within manifold learning is how a new 'unseen' sample can be mapped into a previously learnt embedding. Consider as an example a simple classification problem involving a set of training samples and a seperate set of test samples. We wish to use manifold learning to reduce the dimensionality of these data sets so that we can perform the classification in the lower-dimensional space. The two options open are, either to combine the training and test sets into one and perform manifold learning on this combined dataset before splitting them again in the low-dimensional space, or to run the manifold learning algorithm on the training set and then apply what has been learnt from this manifold learning process to map the test set into the low-dimensional space. The advantage of the latter approach is that it not only potentially less computationally expensive but it also means that new samples can be continually added to the low-dimensional embedding without the need to re-compute the low-dimensional manifold every time. This approach is commonly referred to as the out-of-sample extension. It is worth noting that the out-of-sample extension problem can appear similar in many ways to the problem of incremental learning (Law and Jain 2006), where the low-dimensional manifold is incrementally learnt over a number of iterations of new samples being inserted. This is different from the out-of-sample problem where a new sample simply needs to be mapped into the low-dimensional space without affecting the low-dimensional manifold and requiring a re-learning or change in the manifold parameterization for future learning.

Many existing manifold learning techniques do not naturally contain an out-of-sample extension so research has been undertaken to find ways of extending manifold learning techniques to handle new samples. Bengio et al. (Bengio et al. 2003) presented ways of extending some well known manifold learning techniques: ISOMAP (Tenenbaum, de Silva, and Langford 2000), Locally Linear Embeddings (Roweis and Saul 2000), Laplacian Eigenmaps (Belkin and Niyogi 2003) and Multidimensional Scaling (Cox and Cox 2001), to handle the out-of-sample extension problem. The framework developed by Bengio et alṙelies on phrasing the out-of-sample problem as a kernel problem where a continuous kernel function is defined in order to generalize the existing embeddings to a new data point. Other approaches have been presented that attempt to extend specific manifold learning algorithms to handle new samples (e.g. Maximum Variance Unfolding (Weinberger and Saul 2006; Chin and Suter 2008), LTSA & LLE (Zhang and Zha 2005; Saul and Roweis 2003)) but at present there is little work on creating a generalized solution to the out-of-sample problem. Recently Yang et al. (Yang et al. 2010) proposed a

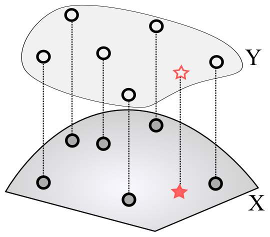


Figure 1: The new sample is attached to its nearest neighborhood in the high-dimensional space and then projected onto the low-dimensional hyperplane defined by the principal components of that neighborhood.

method for a generalized out-of-sample method based on their manifold learning technique Local and Global Regressive Mapping. Regularization is used to learn a model to allow out-of-sample extrapolation and as such they claim that their framework can be applied to any manifold learning algorithm to enable an out-of-sample extension.

In this paper we present a generalized out-of-sample extension (GOoSE) solution. Unlike existing approaches we do not require information to be retained from the learning process, such as the pairwise distance matrix or the resultant eigenvectors, we simply learn the mapping from the original high-dimensional data and its low-dimensional counterpart. As such our method is independent of any specific manifold learning algorithm. The change in local geometry between the high and low-dimensional spaces provides the information needed to compute the transformation of new samples into the low-dimensional space. This simplicity means that our approach can be used to extend any manifold learning technique to handle the out-of-sample extension problem.

The rest of this paper is structured as follows. We begin by outlining the algorithm behind our generalized solution before moving on to show how this generalized solution performs on artificial and real world data. In the Results section we show how using GOoSE can produce comparable results to the existing out-of-sample techniques described above. We end by presenting conclusions and possible directions for future work.

Algorithm

The basic premise of our algorithm is to find the transformation that maps a new unseen sample's neighborhood from the high-dimensional to the low-dimensional spaces. This transformation is equivalent, as far as possible, to running the manifold learning technique on the given sample.

Given the original data set $\mathbf{X} = \{x_i\}_{i=1}^{n} \in \mathbb{R}^p$ and its low-dimensional representation $\mathbf{Y} = \{y_i\}_{i=1}^{n} \in \mathbb{R}^q$, where $q \ll p$, we wish to find the low-dimensional approximation, $\varphi \in \mathbb{R}^q$, of an unkown sample, $\phi \in \mathbb{R}^p$, given that $\phi \ni \mathbf{X}$. We assume that $\mathbf{X}$ is sampled from a hidden manifold $\mathcal{M}$, that is $\mathbf{X} \subseteq \mathcal{M}$, and also that at a local scale $\mathcal{M}$ is linear (i.e. $\mathcal{M}$ is a C^∞ manifold). Since $\mathbf{Y}$ is the result of manifold learning we can describe $\mathbf{Y}$ in terms of a function on $\mathbf{X}$. That is

$$\mathbf{Y} = f(\mathbf{X}) \tag{1}$$

Unless we are dealing with a linear manifold learning algorithm such as Principal Components Analysis this function will be difficult to learn at a global scale. Instead we can think of $\mathbf{Y}$ as being built up by individual functions for each sample. That is for the i-th datapoint $y_i = f_i(x_i)$. The out-of-sample problem can thus be thought of as finding a function that best approximates the transformation undergone via manifold learning, that is for an unlearnt sample $\min(||\varphi - \varphi'||)$ where φ is the actual embedding of the sample and φ' is its estimated embedding. This problem is evidently cyclical as we need the actual embedding to be able to find the function to minimize but we need to minimize the function to find the actual embedding.

To solve this problem it is helpful to take a step back and consider the situation where we know the actual low-dimensional representation, y, of a sample x. To re-create the embedding of x we can examine the local geometric structure around x in the high and low-dimensional spaces. If we assume that the result of running a manifold learning algorithm is a local change in the neighboring geometry of a sample then we can reformulate the problem as that of finding a simple linear transformation

$$y = \mathbf{AV}x \tag{2}$$

where $\mathbf{A}$ is a similarity transformation matrix and $\mathbf{V}$ is a matrix that projects x into the low-dimensional space. We now seek to find $\mathbf{A}$ and $\mathbf{V}$ that best approximates the local transformation. Given that we know the target dimensionality, q, and we take the manifold to be locally linear we can find the projection matrix $\mathbf{V}$ by performing Principal Components Analysis on a local neighborhood of the k-nearest samples according to Euclidean distance around x. We denote the samples in this neighborhood as $\mathbf{X}_{\mathcal{N}_i}$ and so the principal components are found by

$$\Lambda\mathbf{V} = \mathbf{CV} \tag{3}$$

where $\mathbf{C}$ is the covariance matrix of $\mathbf{X}_{\mathcal{N}_i}$ and $\mathbf{V}$ is a matrix containing as columns the top q-dimensional eigenvectors sorted according to their associated eigenvalues, Λ. We can now find the low-dimensional representation of x by projecting onto the eigenvectors, $y = \mathbf{V}x$ (Figure 1).

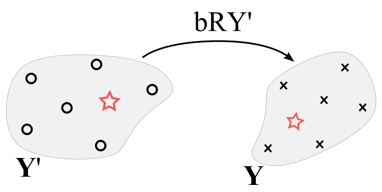


Figure 2: Since we assume that the transformation undergone as a result of manifold learning can be approximated as a local linear transform we aim to find that transform. By applying that transform to the new sample we can find its approximate low-dimensional image.

To find the similarity transformation matrix we need to examine the change in local geometry. We first need to project the local neighborhood $\mathbf{X}_{\mathcal{N}_i}$ into the lower-dimensional space by projecting onto the eigenvectors $\mathbf{V}$ (3). We represent this low dimensional projected neighborhood as $\mathbf{Y}'$ and the same neighborhood of points in the low-dimensional embedding as $\mathbf{Y}$. For convenience we drop the subscripted $\mathcal{N}_i$ so when referring to $\mathbf{Y}$ we actually mean $\mathbf{Y}_{\mathcal{N}_i}$ and similary $\mathbf{Y}'$ is $\mathbf{Y}'_{\mathcal{N}_i}$.

We know that

$$\mathbf{Y} \approx \mathbf{A}\mathbf{Y}' \tag{4}$$

and that the transformation matrix $\mathbf{A}$ can be represented in terms of a seperate scale and rotation component

$$\mathbf{Y} \approx \mathbf{B}\mathbf{R}\mathbf{Y}' \tag{5}$$

where $\mathbf{B}$ is a non-isomporphic scale matrix and $\mathbf{R}$ is the rotation matrix. The task now becomes to find the scale and rotation that transforms $\mathbf{Y}'$ to $\mathbf{Y}$ (Figure 2).

To find the solution to this problem we use a method from statistical shape theory and find the singular value decomposition (SVD) of the matrix $\mathbf{Y'}^T\mathbf{Y}$.

To find the rotation matrix $\mathbf{R}$ and the scale value $\mathbf{b}$ we first find the singular value decomposition

$$\mathbf{Y}'^T\mathbf{Y} = \mathbf{U}\Sigma\mathbf{V}^T \tag{6}$$

the rotation matrix can then be found by

$$\mathbf{R} = \mathbf{U}\mathbf{V}^T \tag{7}$$

Once the rotation has been applied we find the scale matrix by

$$\mathbf{B} = \begin{bmatrix} \frac{max(\mathbf{Y}^1)-min(\mathbf{Y}^1)}{max(\mathbf{Y}'^1)-min(\mathbf{Y}'^1)} & \cdots & \cdots \\ \vdots & \ddots & \vdots \\ \cdots & \cdots & \frac{max(\mathbf{Y}^q)-min(\mathbf{Y}^q)}{max(\mathbf{Y}'^q)-min(\mathbf{Y}'^q)} \end{bmatrix} \tag{8}$$

where $\mathbf{Y}^1$ indicates the column vector containing all samples along the first dimension of $\mathbf{Y}$ and $\mathbf{Y}^q$ indicates the column vector containing all samples along the q^{th} dimension.

Now we return to the original problem of finding the low-dimensional representation, φ, of an unlearnt sample ϕ. As we have shown above, an approximation of the low-dimensional embedding of a neighborhood in the high-dimensional space can be found. So a simple solution to finding the low-dimensional representation of an unlearnt sample is to find the rotation and scale transformations of the sample's nearest neighbors and then applying these transforms to the unlearnt samples. This can be done by finding the k-nearest neighbors of ϕ in $\mathbf{X}$, $\mathbf{X}_{\mathcal{N}_\phi}$. We then find the projection matrix of $\mathbf{X}_{\mathcal{N}_\phi}$ according to (3) and the rotation and scale values according to (7) and (8). The low-dimensional representation, φ, of ϕ then becomes

$$\varphi = \mathbf{B}\mathbf{R}\mathbf{V}_{\mathcal{N}_\phi}\phi \tag{9}$$

This process is described in algorithmic form in Algorithm 1.

Algorithm 1 Generalized Out-of-Sample Extension

```
Require: $x \in \mathbb{R}^D, \mathbf{X} \in \mathbb{R}^D, \mathbf{Y} \in \mathbb{R}^d, k \ll |\mathbf{X}|$
 1: $idx \leftarrow \text{nn}(x, \mathbf{X}, \mathbf{Y}, k)$
 2: $\Lambda\mathbf{V} = \mathbf{C}_\mathbf{X}\mathbf{V}$
 3: $\mathbf{Z}' = \mathbf{X}_{idx}\mathbf{V}_{1\ldots d}$
 4: $(\mathbf{U}\Sigma\mathbf{V}) \leftarrow \text{svd}(\mathbf{Z}'\mathbf{Y}_{idx})$
 5: $\mathbf{B} \leftarrow \text{eye}(d, d)$
 6: $\text{diag}(\mathbf{B}) = [\frac{\text{range}(\mathbf{Y}^1_{idx})}{\text{range}(\mathbf{Z}^1_{idx})} \cdots \frac{\text{range}(\mathbf{Y}^d_{idx})}{\text{range}(\mathbf{Z}^d_{idx})}]$
 7: $\mathbf{T} \leftarrow \mathbf{U}\mathbf{V}^T$
 8: $y \leftarrow x\mathbf{V}_{1\ldots d}$
 9: $y \leftarrow \mathbf{B}y\mathbf{T}$
10: return $y$
```

Discussion & Results

In this section we provide both visual and quanititative evaluation of our method. We begin by defining an embedding error which can be used to analyse the performance of an out-of-sample extension algorithm.We then move on to discuss how GOoSE's only parameter, k, affects the accuracy of the estimated low-dimensional embedding before finally displaying results using both artificial data as well as real world image data.

Embedding Error

To be able to analyse the performance of out-of-sample extensions we need to first define an embedding error. Given a dataset $\mathbf{D}$ we create a training set, $\mathbf{B}$, and test set, $\mathbf{C}$, such that $\mathbf{B} \cup \mathbf{C} = \mathbf{D}$, $\mathbf{B} \cap \mathbf{C} = \emptyset$ and $|\mathbf{B}| = |\mathbf{D}| - |\mathbf{C}|$. As in (Yang et al. 2010) we can obtain the low-dimensional embedding, $\mathbf{Y}$, by running a manifold learning algorithm on the entire dataset $\mathbf{D}$. We can then express $\mathbf{Y}$ as $\mathbf{Y} = [\mathbf{Y}_{\text{train}}, \mathbf{Y}_{\text{test}}]^T$ where $\mathbf{Y}_{\text{train}}$ and $\mathbf{Y}_{\text{test}}$ are the low-dimensional embeddings of the training and test data. Once $\mathbf{Y}$ is know we can use $\mathbf{B}$ to obtain the training set of the manifold and then use an out-of-sample extension method to estimate the low-dimensional embedding of $\mathbf{C}$. We denote the estimated low-dimensional embedding of the test data $\mathbf{Y}'_{\text{test}}$, we can now define an embedding error based on the root mean square error between the actual and estimated test sets

$$e = \sqrt{\frac{\sum(\mathbf{Y}_{\text{test}} - \mathbf{Y}'_{\text{test}})^2}{n}} \tag{10}$$

where n is the number of elements in the test set and both $\mathbf{Y}_{\text{test}}$ and $\mathbf{Y}'_{\text{test}}$ are transformed according to the rotational difference between $\mathbf{Y}_{\text{train}}$ and $\mathbf{B}$ to remove the effect of the manifold learning algorithms mapping the datasets into different low-dimensional spaces[1]. This error measure now

[1]This is something that is not considered by Yang et al. in (Yang et al. 2010) but without this step the results obtained are meaningless as the two low-dimensional embeddings are in different coordinate spaces

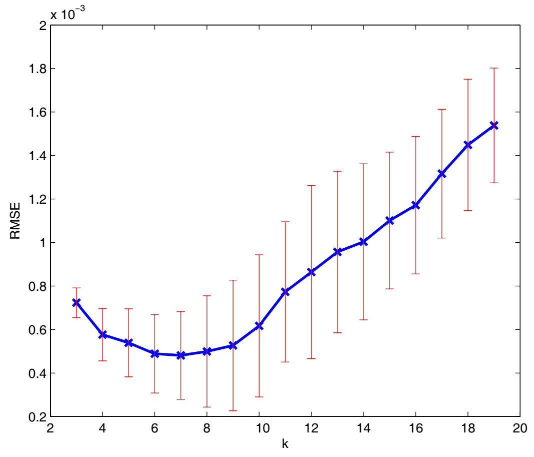


Figure 3: The effect of the neighborhood size parameter k on the embedding error of a dataset with a known low-dimensional manifold.

provides us with a basis of analysing the performance of an out-of-sample extension method, with a low value of e signifying that the estimated test embedding is closer to the actual test embedding than that of a test embedding with a high value of e.

Parameter Selection

Our algorithm has only one free parameter, the neighborhood size k. To test how this parameter affects the performance we ran a set of experiments on a known manifold with a known low-dimensional embedding. We used the Swiss Roll manifold with 2000 samples and the low-dimensional embedding learnt by LTSA (we could have used any manifold learning algorithm but LTSA produces the most faithful result as shown in Figure 4). The data was randomly split into training and test sets with each set having a size of 1000. For each permutation of training and test we used the GOoSE algorithm to try and embed the test set into the low-dimensional space with varying parameters of k within the range $[3, 19]$. The RMSE of the test data for each value of k was recorded and averaged over a series of 10 runs.

Figure 3 shows the results of this test. The graph is shown with associated error bars indicating the standard deviation of the results per value of k. The results show that a minima is reached around $k = 7 \pm 2$, after this point the RMSE increases along with the standard deviation meaning that results obtained with a larger value of k are more unstable. Although this optimum value of k will change depending on what dataset is used, experiments do show that a local minima will always exist. Since the GOoSE algorithm is fast to run it is easy to find an optimum value of k by performing a simple parameter search.

Results

To test our algorithm we use 3 main datasets: a 3-dimensional Swiss Roll, a moving image dataset and the ISOMAP faces data. Each of these datasets presents a different challenge for a manifold learning algorithm and subsequently an out-of-sample extension algorithm.

Swiss Roll The Swiss Roll dataset consists of a 2-dimensional manifold embedded within $\mathbb{R}^3$. This 2-dimensional manifold is a highly-curved plane that is rolled up to resemble a Swiss Roll (Figure 4). A manifold learning algorithm should be able to 'unwrap' this Swiss Roll and embed it into $\mathbb{R}^2$. We used 2000 points sampled from the Swiss Roll and this was randomly split into 1000 samples for training and 1000 samples for test. We used four different manifold learning algorithms (LLE (Roweis and Saul 2000), LTSA (Zhang and Zha 2005), Eigenmaps (Belkin and Niyogi 2003) and LGRM (Yang et al. 2010)) to learn the low-dimensional training embedding before applying GOoSE to estimate the test set's low-dimensional embedding. These algorithms were chosen due to the fact that they all either inherently contain, or have been extended to cope with, the out-of-sample extension problem. For LLE, LTSA and Eigenmaps the neighborhood size parameter was set to 8 and for LGRM we used the parameters shown in (Yang et al. 2010). The results of running our algorithm on the Swiss Roll dataset using the GOoSE k parameter of $k = 7$ are shown in Figure 4. The top row shows the 1000 training samples and the bottom row shows the results of running GOoSE on the test samples. In all cases GOoSE is able to embed the novel samples within the trained manifold to obtain a meaningful embedding of the test set. It is worth noting that the failure of Laplacian Eigenmaps, and to some extent LLE, to produce meaningful low-dimensional embeddings is due to the fact that the problem is under-sampled. As such these techniques are unable to build an adequate model of the manifold from the training set leading to an incorrect low-dimensional embedding. However, this does enable us to show that even in the case of a distorted embedding GOoSE is able to embed novel samples according to the shape of the trained embedding.

To obtain quantative analysis of our algorithm and to compare it against existing approaches we measured the embedding error of our algorithm using a 10 fold cross validation approach. The data was randomly split into 10 folds with 9 being used for training and 1 for test. This was repeated until all folds had been used as a test set. A manifold learning algorithm was then used to obtain the full low-dimensional embedding as well as the training set's low-dimensional embedding. For each run the RMSE was recorded when using GOoSE and also when using the given manifold learning algorithm's out-of-sample extension. For LLE and LTSA we used the out-of-sample approach outlined in (Saul and Roweis 2003); for Eigenmaps we used the approach outlined in (Bengio et al. 2003) and we used LGRM's built in approach (Yang et al. 2010). Thus for each run of the experiment using a given manifold learning algorithm we obtain two different error scores: the error obtained from using the out-of-sample extension associated with the given algorithm

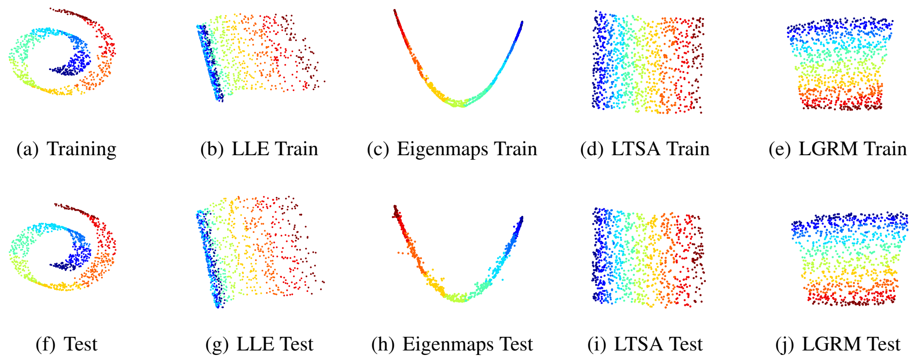


Figure 4: Results of running the GOoSE algorithm on the Swiss Roll data with 1000 samples used from training and 1000 samples used for test. GOoSE's k parameter was set to $k = 7$. The neighborhood size parameter for LLE, LTSA and Eigenmaps was set to 8 while the parameters for LGRM were set according to (Yang et al. 2010).

(the default method) and the error obtained from running GOoSE as the out-of-sample method. For each test GOoSE was run multiple times with differing values of k and the minimum RMSE was taken. The averaged results are shown in the graph in Figure 5. When compared with existing out-of-sample approaches our algorithm is able to consistently out perform current methods. The average RMSE across all methods for GOoSE is $e = 0.0002$ when compared with $e = 0.0043$ for using the algorithms' built in out-of-sample methods. There is also large variation in the performance of existing out-of-sample methods, $\sigma = 0.0044$, with LLE performing the worst ($e = 0.010$) and LGRM performing the best ($e = 0.0008$). The variation between different manifold learning algorithms when using GOoSE is considerably lower, $\sigma = 0.0001$. This shows the stability of GOoSE and its effectiveness regardless of what manifold learning algorithm is used to learn the low-dimensional embedding.

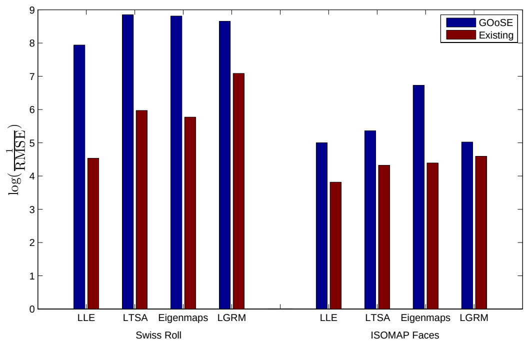


Figure 5: Average embedding quality for each of the out-of-sample extension algorithms on the Swiss Roll and ISOMAP faces datasets. Due to the large range in results the plot is shown with the Y axis scaled as $\log(\frac{1}{RMSE})$, meaning a larger value indicates a better quality embedding.

Moving Image To test our algorithm on image data we use two different datasets the first of which consits of an image moving across a black background. The dataset contains 4096 images of size 96×96 pixels and so the high-dimensional data lies in $\mathbb{R}^{9216}$. The training data consists of 2048 randomly selected samples and the remaining samples are used as test. Local Tangent Space Alignment (Zhang and Zha 2005) with parameter $k = 8$ was used to learn the low-dimensional embedding of the training set as it was able to find a meaningful low-dimensional embedding of the data. Figure 6 shows the resulting 2-dimensional embedding with the training data indicated by blue dots (●) and the test data indicated by a red plus-signs (+). As can be seen the test samples fit nicely into the 'gaps' of the training data as would be expected. The data consists of dense regions of samples around the corners and a sparser region of samples in the center (this is due to the manifold being curved at the edges and so is not truely 2-dimensional). Even with the more sparsely sampled central region GOoSE manages to place the unlearnt samples into the correct regions.

ISOMAP Faces The second image dataset used is the ISOMAP faces dataset (Tenenbaum, de Silva, and Langford 2000) consisting of a set of 698 faces in $\mathbb{R}^{4096}$ under different pose and illumination conditions. This dataset is interesting as it has intrinsic dimensionality of 4 (Kégl 2002) meaning the quality of results are not visually assessable. Therefore our algorithm's performance on this dataset along with the performance of other out-of-sample methods is shown in Figure 5. The neighborhood size parameters for LLE, LTSA and Eigenmaps were set to 8 while the parameters for LGRM were set according to (Yang et al. 2010). GOoSE was run multiple times with differing values of k and the minimum RMSE was taken. As with the Swiss Roll dataset we used a 10 fold cross validation approach and averaged the results (the full details are described in the Swiss Roll section above). Again GOoSE is able to outperform existing out-of-sample techniques. The average embedding error for GOoSE on the ISOMAP faces data is $e = 0.0048$ with

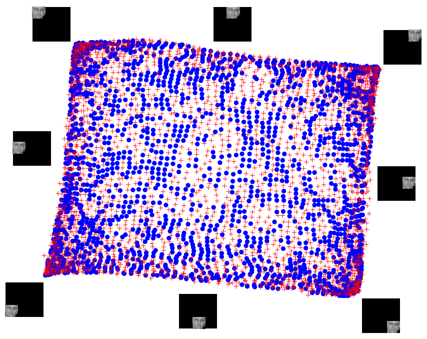


Figure 6: Result of running the GOoSE algorithm on high-dimensional image data. The data consists of 4096 images of size 92×92 split randomly using half as training and half as test. The blue dots represent the training samples and the red plus-signs represent the samples learnt using GOoSE.

standard deviation of $\sigma = 0.0026$. For the existing out-of-sample methods the average error is $e = 0.0144$ with standard deviation of $\sigma = 0.0052$. Although the standard deviation of the results from the GOoSE algorithm on this dataset is higher it is still able to consistently out perform existing out-of-sample methods.

Conclusions & Future Work

We have presented a novel and simple technique to solve the generalized out-of-sample extension problem in manifold learning. Our algorithm, GOoSE, applies the local geometric change between neighborhoods in the high and low-dimensional space to any unlearnt sample to obtain its low-dimensional embedding. The method works by learning the transformation that maps the neighborhood of the unlearnt sample from the high to the low-dimensional space. This transformation is then applied to the new sample to obtain an estimation of its low-dimensional embedding.

The results show that this method is able to succesfully embed new datapoints into non-linear manifolds. We have shown that the GOoSE algorithm is able to embed new samples into previously learnt manifolds regardless of the manifold learning technique used. GOoSE also significantly outperforms existing out-of-sample techniques when tested on artificial and real world data. This make GOoSE a powerful and versatile tool for statistical learning as it is indepenent of the manifold learning technique used and only requires access to the original data and the learnt low-dimensional embedding.

We are currently working on a version of the GOoSE algorithm that reverses the out of sample process, that is it aims to solve the pre-image problem (given a sample in the low-dimensional space we wish to find its high-dimensional image). Using similar methodology outlined in this paper we aim to produce a pre-image algorithm that can be combined with the GOoSE algorithm to form a framework for easily mapping between the high and low-dimensional spaces.

References

Belkin, M., and Niyogi, P. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation* 15(6):1373–1396.

Bengio, Y.; Paiement, J.-F.; Vincent, P.; Delalleau, O.; Roux, N. L.; and Ouimet, M. 2003. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *Advances in Neural Information Processing Systems 16, NIPS 2003*.

Chin, T.-J., and Suter, D. 2008. Out-of-sample extrapolation of learned manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30:1547–1556.

Cox, T. F., and Cox, M. 2001. *Multidimensional Scaling*. Chapman and Hall.

Hotelling, H. 1933. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology* (24):417–441,498–520.

Kégl, B. 2002. Intrinsic dimension estimation using packing numbers. In *Advances in Neural Information Processing Systems 15, NIPS 2002*.

Law, M. H., and Jain, A. K. 2006. Incremental nonlinear dimensionality reduction by manifold learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28:377–391.

Patwari, N.; III, A. O. H.; and Pacholski, A. 2005. Manifold learning visualization of network traffic data. In *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, MineNet '05, 191–196.

Roweis, S. T., and Saul, L. K. 2000. Nonlinear dimensionality reduction by locally linear embedding. *SCIENCE* 290:2323–2326.

Saul, L. K., and Roweis, S. T. 2003. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *J. Mach. Learn. Res.* 4:119–155.

Strange, H., and Zwiggelaar, R. 2009. Classification performance related to intrinsic dimensionality in mammographic image analysis. In *Proceedings of the Thirteenth Annual Conference on Medical Image Understanding and Analysis*, 219–223.

Tenenbaum, J. B.; de Silva, V.; and Langford, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *SCIENCE* 290:2319–2323.

Verbeek, J. 2006. Learning nonlinear image manifolds by global alignment of local linear models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28(8):1236–1250.

Weinberger, K. Q., and Saul, L. K. 2006. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, 1683–1686.

Yang, Y.; Nie, F.; Xiang, S.; Zhuang, Y.; and Wang, W. 2010. Local and global regressive mapping for manifold learning with out-of-sample extrapolation. In *Twenty-Fifth American Association for Artificial Intelligence Conference 2010, (AAAI-2010).*

Zhang, Z., and Zha, H. 2005. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM J. Sci. Comput.* 26:313–338.

Bibliography

W. N. Anderson and T. D. Morley. Eigenvalues of the Laplacian of a graph. *Linear and Multilinear Algebra*, 18:141–145, 1985.

M. Aupetite. Learning Topology with the Generative Gaussian Graph and the EM algorithm. In *Advances in Neural Information Processing Systems 18: Proceedings of the 2006 Conference (NIPS)*, pages 83–90, 2006.

G. Bakir, J. Weston, and B. Schölkopf. Learning to Find Pre-Images. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2004 Conference (NIPS)*, 2004.

M. Balasubramanian and E. L. Schwartz. The Isomap algorithm and topological stability. *Science*, 295:7a (Techincal Comments), 2002.

M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14: Proceedings of the 2002 Conference (NIPS)*, pages 585–591. MIT Press, 2002.

R. Bellman. *Adaptive Control Processes: A Guided Tour.* Princeton University Press, 1961.

Y. Bengio, O. Delalleau, N. L. Roux, J. F. Paiement, P. Vincent, and M. Ouimet. Learning eigenfunctions links spectral embedding and Kernel PCA. *Neural Computing*, 16(10):2197–2219, 2004.

Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet. Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. In *Advances in Neural Information Processing Systems 15: Proceedings of the 2003 Conference (NIPS)*, pages 177–184, 2003.

K. P. Bennett, P. S. Bradley, and A. Demiriz. Constrained k-means clustering. Technical Report MSR-TR-2000-65, Microsoft Research, May 2000.

M. Bernstein, V. de Silva, J. C. Langford, and J. B. Tenenbaum. Graph approximations to geodesics on embedded manifolds. Technical report, Department of Psychology, Stanford University, 2000.

C. M. Bishop, M. Svensen, and C. K. I. Williams. Gtm: The generative topographic mapping. *Neural Computation*, 10:215–234, 1998.

T. S. Blyth and E. F. Robertson. *Basic Linear Algebra*. Springer, 2 edition, 2007.

C. A. Bouman. Cluster: An unsupervised algorithm for modeling Gaussian mixtures. Available from http://www.ece.purdue.edu/~bouman (Link Checked: 26/07/2011), April 1997.

M. Brand. Charting a manifold. In *Advances in Neural Information Processing Systems 15: Proceedings of the 2003 Conference (NIPS)*, pages 961–968. MIT Press, 2003.

M. Brand. From subspace to submanifold methods. In *Proceedings of the 15th British Machine Vision Conference*, 2004.

A. Brun, C. F. Westin, M. Herberthson, and H. Knutsson. LOGMAP: preliminary results using a new method for manifold learning. In *Proceedings of the SSBA Symposium on Image Analysis*, 2005.

C. J. C. Burges. Geometric methods for feature extraction and dimensional reduction: A guided tour. Technical Report MSR-TR-2004-55, Microsoft Research, 2004.

F. Camastra and A. Vinciarelli. Estimating the intrinsic dimension of data with a fractal-based approach. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 24(10):1404–1407, 2002.

M. P. d. Carmo. *Riemannian Geometry.* Birkhauser, 1992.

M. A. Carreira-Perpinán. Proximity graphs for clustering and manifold learning. In *Advances in Neural Information Processing Systems 17: Proceedings of the 2005 Conference (NIPS)*, pages 225–232. MIT Press, 2005.

T.-J. Chin and D. Suter. Out-of-sample extrapolation of learned manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(9): 1547–1556, September 2008.

K. W. Chon, Y. Han, and W. Tak. Concurrent threads and optimal parallel minimum spanning trees algorithms. *Journal of the Association for Computing Machinery*, 48(2):297–323, 2001.

D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.

J. A. Costa and A. O. Hero. Geodesic entropic graphs for dimension and entropy estimation in manifold learning. *IEEE Transactions on Signal Processing*, 52 (8):2210–2221, 2004.

T. F. Cox and M. A. A. Cox. *Multidimensional Scaling.* Chapman and Hall, 2001.

K. I. Diamantaras and S. Y. Kung. *Principal component neural networks: theory and Principal Component Neural Networks: Theory and Applications.* Adaptive and Learning Systems for Signal Processing, Communications and Control. Wiley-Interscience, 1996.

E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. In *Proceedings of the National Academy of Science of the United States of America*, volume 100, 2003.

M. Fan, H. Qiao, and B. Zhang. Intrinsic dimension estimation of manifolds by incising balls. *Pattern Recognition*, 42:780–787, 2009.

I. K. Fodor. A survery of dimension reduction techniques. Technical Report UCRL-ID-148, US Department of Energy, 2002.

K. Fukunaga and D. R. Olsen. An algorithm for finding the intrinsic dimensionality of data. *IEEE Transactions on Computers*, C-20(2):176–193, 1971.

Z. Gao and J. Liang. The dynamical neighborhood selection based on the sampling density and manifold curvature for isometric data embedding. *Pattern Recognition Letters*, 32:202–209, 2011.

V. Garcia, E. Debreuve, and M. Barlaud. Fast k nearest neighbor search using GPU. *Computer Vision and Pattern Recognition Workshop*, pages 1–6, 2008.

M. Gashler, D. Ventura, and T. Martinez. Iterative non-linear dimensionality reduction by manifold sculpting. In *Advances in Neural Information Processing Systems 20: Proceedings of the 2008 Conference (NIPS)*, 2008.

X. Geng, D.-C. Zhan, and Z.-H. Zhou. Supervised nonlinear dimensionality reduction for visualization and classification. *IEEE Transactions on Systems, Man and Cybernetics*, 35(6):1098–1107, 2005.

S. Gerber, T. Tasdizen, and R. Whitaker. Robust non-linear dimensionality reduction using successive 1-dimensional Laplacian Eigenmaps. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

Y. Goldberg and Y. Ritov. Local procrustes for manifold embedding: a measure of embedding quality and embedding algorithms. *Machine Learning*, 77(1): 1–25, 2009.

Y. Goldberg, A. Zakai, D. Kushnir, and Y. Ritov. Manifold learning: The price of normalization. *Journal of Machine Learning Research*, 9(1909-1939), 2008.

J. C. Gower. Generalized Procrustes Analysis. *Psychometrika*, 40(1):33–51, 1975.

P. Grassberger and I. Procaccia. Measuring the strangeness of strange attractors. *Physica*, D9(189):189–208, 1983.

S. I. Grossman. *Elementary Linear Algebra*. Brooks/Cole, 5th edition, 1994.

O. Grygorash, Y. Zhou, and Z. Jorgensen. Minimum spanning tree based clustering algorithms. In *ICTAI '06 Proceedings of the 18th IEEE Conference on Tools with Artificial Intelligence*, 2006.

I. Guyon and A. Elisseeff. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, 3:1157–1182, March 2003.

J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. Technical Report TR-110, Max Planck Institut für biologische Kybernetik, July 2003.

A. J. Hanson. *Graphics Gems IV*. Academic Press Professional, Inc., San Diego, CA, USA, 1994.

W. He, E. R. E. Denton, and R. Zwiggelaar. Mammographic image segmentation and risk classification using a novel texture signature based methodology. In *Digital Mammography / IWDM*, pages 526–533, 2010.

X. He, S. Yan, Y. Hu, P.Niyogi, and H. J. Zhang. Face recognition using laplacianfaces. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 27(3):328–340, 2005.

X. He and P. Niyogi. Locality Preserving Projections. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2004 Conference*, pages 153–160. MIT Press, 2004.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006 2006.

G. Hinton and S. Roweis. Stochastic Neighbor Embedding. In *Advances in Neural Information Processing Systems 15: Proceedings of the 2003 Conference (NIPS)*, pages 833–840. MIT Press, 2000.

H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

R. Huber, H. Ramoser, K. Mayer, H. Penz, and M. Rubik. Classification of coins using an eigenspace approach. *Pattern Recognition Letters*, 26(1):61–75, 2005.

P. Jia, J. Yin, X. Huang, and D. Hu. Incremental Laplacian Eigenmaps by preserving adjacent information between data points. *Pattern Recognition Letters*, 30:1457–1463, 2009.

V. John, E. Trucco, and S. McKenna. Markerless human motion capture using charting and manifold constrained particle swarm optimisation. In B. Tiddeman and H. Strange, editors, *Proceedings of the BMVC 2010 UK postgraduate workshop*, 2010.

G. Jun and J. Ghosh. Nearest-manifold classification with Gaussian processes. In *In Proceedings of the International Conference on Pattern Recognition*, pages 914–917, 2010.

N. Kambhatla and T. K. Leen. Dimension reduction by local Principal Components Analysis. *Neural Computation*, 9(7):1493–1516, 1994.

K. Karhunen. Zur spektraltheorie stochastischer prozesse. *Annales Academiæ Sciientiarum Fennicæ*, 34, 1946.

Z. Karina, G. Gilles, and C. Stephane. Estimation of tangent planes for neighborhood graph corrections. In *ESANN'2007 Proceedings of the European Symposium on Artificial Neural Networks*, pages 25–27, 2007.

B. Kégl. Intrinsic dimension estimation using packing numbers. In *Neural Information Processing Systems*, 2002.

K. I. Kim, K. Jung, and H. J. Kim. Face recognition using kernel principal components analysis. *IEEE Signal Processing Letters*, 9(2):40–42, 2002.

J. M. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, STOC '97, 1997.

T. Kohonen. *Self-Organizing Maps.* New York: Springer-Verlag, 1995.

O. Kouropteva, O. Okun, and M. Pietikainen. Selection of the optimal parameter value for the locally linear embedding algorithm. In *Proceedings of the First International Conference on Fuzzy Systems and Knowledge Discovery*, pages 359–363, 2002.

J. Kwok and I. Tsang. The pre-image problem in kernel methods. *IEEE Transactions on Neural Networks*, 15:1517–1525, 2004.

S. Lafon and A. B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1393–1403, 2006.

M. Law and A. Jain. Incremental nonlinear dimensionality reduction by manifold learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):377–391, March 2006.

J. A. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Springer, 2007.

J. A. Lee and M. Verleysen. Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomputing*, 72:1431–1443, 2009.

J. A. Lee, A. Lendasseb, and M. Verleysen. Nonlinear projection with curvilinear distances: Isomap versus curvilinear distance analysis. *Neurocomputing*, 57: 49–76, 2004.

J. A. Lee and M. Verleysen. Nonlinear dimensionality reduction of data manifolds with essential loops. *Neurocomputing*, 67:29–53, 2005.

E. Levina and P. J. Bickel. Maximum likelihood estimation of intrinsic dimension. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2004 Conference (NIPS)*, 2004.

M. Lewandowski, D. Markis, and J. C. Nebel. Automatic configuration of spectral dimensionality reduction methods. *Pattern Recognition Letters*, 31:1720–1727, 2010.

A. Lima, H. Zen, Y. Nankaku, C. Miyajima, K. Tokuda, and T. Kitamura. On the use of Kernel PCA for feature extraction in speech recognition. *IEICE Transactions on Information Systems*, E87-D(12):2802–2811, 2004.

B. Lin, X. He, Y. Zhou, L. Liu, and K. Lu. Approximately Harmonic Projection: Theoretical analysis and an algorithm. *Pattern Recognition*, 43:3307–3313, 2010.

T. Lin, H. Zha, and S. U. Lee. Riemannian manifold learning for nonlinear dimensionality reduction. In *In Proceedings of ECCV*, 2006.

C. B. Liu, R. S. Lin, N. Ahuja, and M. H. Yang. Dynamic textures synthesis as nonlinear manifold learning and traversing. In *In Proceedings of the British Machine Vision Conference (BMVC)*, 2006a.

X. Liu, J. Yin, Z. Feng, and J. Dong. Incremental manifold learning via tangent space alignment. In *ANNNPR*, pages 107–121, 2006b.

M. Loéve. Fonctions aléatoire du second odre. *Processus stochastiques et movement Brownien*, page 299, 1948.

L. v. Maaten and G. Hinton. Visualizing data using t-SNE. Technical Report UTML TR 2008-001, University of Toronto, May 2008.

L. v. Maaten, E. Postma, and J. van den Herik. Dimensionality reduction: A comparitive review. Technical Report TiCC-TR 2009-005, Tilburg University, 2009. Unpublished.

K. Mardia, J. Kent, and J. Bibby. *Multivariate Analysis*. New York: Academic, 1979.

B. Mohar. The Laplacian spectrum of graphs. In *Graph Theory, Combinatorics, and Applications*, pages 871–898, 1991.

K. Nam, H. Je, and S. Choi. Fast stochastic neighbor embedding: A trust-region algorithm. In *Proceedings of the IEEE International Joint Conference on Neaural Networks*, 2004.

M. Nathan and K. T. John. Parameterless Isomap with adaptive neighborhood selection. In *Deutsche Arbeitsgemeinschaft für Mustererkennung DAGM*, 2006.

S. A. Nene and S. K. Nayar. A simple algorithm for nearest neighbour search in high dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):989–1003, September 1997.

S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library (COIL-20). Technical report, Technical Report CUCS-005-96., 1996.

A. Y. Ng, M. I. Jordan, and Y. Weiss. On Spectral Clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14: Proceedings of the 2002 Conference (NIPS)*, 2002.

M. Partridge and R. Calvo. Fast dimensionality reduction and simple PCA. *Intelligent Data Analysis*, 2:203–214, 1998.

K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

K. W. Pettis, T. A. Bailey, A. K. Jain, and R. C. Dubes. An intrinsic dimensionality estimator from near-neighbor information. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 1:25–37, 1979.

J. C. Platt. FastMap, MetricMap and Landmark MDS are all Nyström algorithms. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, pages 261–268, 2005.

N. M. Rajpoot, M. Arif, and A. H. Bhalerao. Unsupervised learning of shape manifolds. In *Proceedings of the British Machine Vision Conference*, 2007.

V. Robins. *Computational Topology at Multiple Resolutions.* PhD thesis, Department of Applied Mathematics at the University of Colorado, Boulder, 2000.

V. Robins, J. D. Meiss, and E. Bradley. Computing Connectedness: Disconnectedness and Discreteness. *Physica D*, 139(3-4), 2000.

S. T. Roweis. EM algorithms for PCA and SPCA. In *Advances in Neural Information Processing Systems 9: Proceedings of the 1997 Conference (NIPS)*, volume 10, pages 626–632, 1997.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by Locally Linear Embedding. *Science*, 290:2323–2326, 2000.

S. Roweis, L. K. Saul, and G. E. Hinton. Global coordination of local linear models. In *Advances in Neural Information Processing Systems 14: Proceedings of the 2002 Conference (NIPS)*. MIT Press, 2002.

S. Roychowdhury and J. Ghosh. Robust Laplacian Eigenmaps using global information. In *Manifold Learning and Applications: Papers from the AAAI Fall Symposium (FS-09-04)*, pages 42–49, 2009.

H. Sahbi. A particular Gaussian Mixture Model for clustering and its application to image retrival. *Soft Computing*, 12(7):667–676, 2008.

O. Samko, A. D. Marshall, and P. Rosin. Selection of the optimal parameter value for the Isomap algorithm. *Pattern Recognition Letters*, 27(9):968–979, 2006.

J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C.18(5):401–409, 1969.

B. Scholkopf, E. Smola, L. Bottou, C. Burges, H. Bultho, K. Gegenfurtner, and P. H. Ner. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

D. W. Scott and J. R. Thompson. Probablitiy density estimation in higher dimensions. In J. R. Gentle, editor, *Proceedings of the Fifteenth Symposium on the Interface*, pages 173–179. Elsevier Science Publishers, 1983.

H. S. Seung and D. D. Lee. The manifold ways of perception. *Science*, 290 (5500):2268–2269, 2000.

F. Sha and L. K. Saul. Analysis and extension of spectral methods for nonlinear dimensionality reduction. In *Proceedings of the 22nd International Conference on Machine Learning*, volume 119 of *ACM Internation Conference Proceeding Series*, pages 784–791, Bonn, Germany, 2005. ACM.

J. Shawe-Taylor and N. Christianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

R. Sibson. Studies in the robustness of multidimensional-scaling: Procrustes statistics. *Journal of the Royal Statistical Society*, 40(2):234–238, 1978.

J. G. Silva, J. S. Marques, and J. M. Lemos. Selecting landmark points for sparse manifold learning. In *Advances in Neural Information Processing Systems 17: Proceedings of the 2005 Conference (NIPS)*, 2005.

V. d. Silva and G. Carlsson. Topological estimation using witness complexes. In *Eurographics Symposium on Point-Based Graphics*, 2004.

V. d. Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems 15: Proceedings of the 2003 Conference (NIPS)*, pages 705–712. MIT Press, 2003a.

V. d. Silva and J. B. Tenenbaum. Unsupervised learning of curved manifolds. *Nonlinear Estimation and Classification, Lecture Notes in Statistics*, 171, 2003b.

R. Souvenir, Q. Zhang, and R. Pless. Image manifold interpolation using free-form deformations. In *Proceedings of the IEEE International Conference on Image Processing*, pages 1437–1440, 2006.

H. Strange and R. Zwiggelaar. Classification performance related to intrinsic dimensionality in mammographic image analysis. In *Proceedings of the Thirteenth Annual Conference on Medical Image Understanding and Analysis*, pages 219–223, 2009.

H. Strange and R. Zwiggelaar. Iterative Hyperplane Merging: A Framework for Manifold Learning. In *Proceedings of the British Machine Vision Conference*, 2010.

V. Strassen. Gaussian elimination is not optimal. *Numerical Mathematics*, 13: 354–546, 1969.

A. Talwalkar, S. Kumar, and H. Rowley. Large-scale manifold learning. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –8, 2008.

J. B. Tenenbaum. Mapping a manifold of perceptual observations. In *Advances in Neural Information Processing Systems 10: Proceedings of the 1998 Conference (NIPS)*, pages 682–688, 1998.

J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2322, 2000.

L. Teng, H. Li, X. Fu, W. Chen, and I. F. Shen. Dimension reduction of microarray data based on local tangent space alignment. In *Proceedings of the 4th IEEE International Conference on Cognitive Informatics*, pages 154–159, 2005.

N. Thorstensen, F. Ségonne, and R. Keriven. Pre-Image as Karcher Mean using Diffusion Maps: Application to Shape and Image Denoising. In *Proceedings of SSVM*, pages 721–732, 2009.

M. E. Tipping. Sparse kernel principal components analysis. In *Advances in Neural Information Processing Systems 12: Proceedings of the 2000 Conference (NIPS)*, volume 13, pages 633–639, 2000.

W. S. Torgerson. Multidimensional Scaling I: Theory and method. *Psychometrika*, 17:401–419, 1952.

M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pages 586–591, 1991.

G. Valiente. *Algorithms on Trees and Graphs.* Springer-Verlag, Berlin Heidelberg, 2002.

J. Venna and S. Kaski. Local Multidimensional Scaling. *Neural Networks*, 19: 889–899, 2006.

J. Verbeek. Learning non-linear image manifolds by global alignment of local linear models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1236–1250, 2006.

P. J. Verveer and R. P. W. Duin. An evaluation of intrinsic dimensionality estimators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1):81–86, 1995.

C. Wang and S. Mahadevan. Manifold alignment using Procrustes analysis. In *Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland*, 2008.

J. Wang, Z. Zhang, and H. Zha. Adaptive manifold learning. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2004 Conference (NIPS)*, 2004.

K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70: 77–90, 2006a.

K. Q. Weinberger and L. K. Saul. An introduction to nonlinear dimensionality reduction by Maximum Variance Unfolding. In *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006b.

K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *In Proceedings of the 21st International Conference on Machine Learning*, pages 839–846. ACM Press, 2004.

Y. Weiss. Segmentation using eigenvectors: A unifying view. In *Proceedings IEEE International Conference on Computer Vision*, pages 975–982, 1999.

E. W. Weisstein. Characteristic Equation. In *MathWorld - A Worlfram Web Resource* `http://mathworld.wolfram.com/CharacteristicEquation.html`, 2011.

G. Wen, L. Jiang, and N. R. Shadbolt. Using graph algebra to optimize neighborhood for isometric mapping. In *20th International Joint Conference on Artificial Intelligence*, pages 2398–2403, 2007.

C. K. I. Williams. On a connection between Kernel PCA and metric multidimensional scaling. *Machine Learning*, 46(1-3):11–19, 2002.

S. Wold, K. Esbensen, and P. Geladi. Principal Components Analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1-3):37–52, 1987.

S. Xiang, F. Nie, Y. Song, C. Zhang, and C. Zhang. Embedding new data points for manifold learning via coordinate propagation. *Knowledge Information Systems*, 19:159–184, 2009.

R. Xu, S. Damelin, and D. C. Wunsch. Application of diffusion maps in gene expression data-based cancer diagnosis analysis. In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4613–4616, 2007.

K.-W. Yang. A basis for the intersection of subspaces. *Mathematics Magazine*, 70(4):297, October 1997.

L. Yang. Building k edge-disjoint spanning trees of minimum total length for isometric data embedding. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 27(10):1680–1683, 2005.

L. Yang. Distance-Preserving Projection of high-dimensional data for nonlinear dimensionality reduction. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 26(9):1243–1246, 2004.

Y. Yang, F. Nie, S. Xiang, Y. Zhuang, and W. Wan. Local and Global Regressive Mapping for manifold learning with out-of-sample extrapolation. In *Proceedings of AAAI-2010*, 2010.

J. Yin, D. Hu, and Z. Zhoi. Growing locally linear embedding for manifold learning. *Journal of Pattern Recognition Research*, 1:1–16, 2007.

P. Zhang, H. Qiao, and B. Zhang. An improved local tangent space alignment method for manifold learning. *Pattern Recognition Letters*, 32:181–189, 2011.

T. Zhang, J. Yang, D. Zhao, and X. Ge. Linear local tangent space alignment and application to face recognition. *Neurocomputing*, 70:1547–1533, 2007.

Z. Zhang and H. Zha. Principal manifolds and nonlinear dimension reduction via local tangent space alignment. *SIAM Journal on Scientific Computing*, 26(1):313–338, 2004.

S. Zhu, C. Guo, Y. Wu, and Y. Wang. What are Textons? In *Proceedings of the 7th European Conference on Computer Vision (ECCV)*, 2002.