Semidefinite Embedding for the Dimensionality Reduction of DNA Microarray Data

by

Rosina Kharal

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Masters of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2006

Abstract

Harnessing the power of DNA microarray technology requires the existence of analysis methods that accurately interpret microarray data. Current literature abounds with algorithms meant for the investigation of microarray data. However, there is need for an efficient approach that combines different techniques of microarray data analysis and provides a viable solution to dimensionality reduction of microarray data. Reducing the high dimensionality of microarray data is one approach in striving to better understand the information contained within the data. We propose a novel approach for dimensionality reduction of microarray data that effectively combines different techniques in the study of DNA microarrays. Our method, ***KAS*** *(kernel alignment with semidefinite embedding)*, aids the visualization of microarray data in two dimensions and shows improvement over existing dimensionality reduction methods such as PCA, LLE and Isomap.

Acknowledgements

I would like to extend sincere appreciation to Forbes J. Burkowski, for his guidance and excellent supervision over the past four years. This work would not be what it is today, without his patience and extensive reviews. It has been truly an honour to work with someone of such integrity and expertise, and I have learned a great deal.

I would like to thank Richard Mann and Brendan McConkey for taking time to read this work and making helpful suggestions.

Also, a special thanks to Kilian Weinberger for his explanations and many conversations regarding the semidefinite embedding algorithm. I would like to thank Shirley Hui, Sadaf and Fawad Chughtai for their helpful comments and consultation as fellow students.

On a personal note, I have many people to thank whom either directly or indirectly contributed to helping me reach the end of this thesis. I am wholeheartedly grateful to my parents for their love and support over the many years of my life, and for raising me to a day where I may formally thank them in this manner. My husband, Omar Nafees, has been a great source of encouragement and guidance throughout the years. I would like to genuinely thank him for his advice and kindness when I really needed it, and for helping so much with everything up until the very last day. I would like to thank my little boys, Abdulwadud and Mustafa, for their innocence and laughter that has made everything worth while.

There are many people that helped me with my children over the years. I thank Fariha Nafees, for always taking the kids out and giving them a great time, and for listening and advising me, and sending her prayers and blessings. My husband's parents and entire family for their love, support and kind words over the telephone. Rashad and Farah Kharal for helping so much with the kids, since the day the kids were born. My two dear friends in Waterloo, Ambreen Syed and Amaal Juma, for never failing to be there when I most needed, and supporting me and assisting me through the difficult times. Ruby Jamal for her kindness and compassion throughout the years. I would like to thank Asaf Kharal for reviewing this thesis and sending helpful comments, and for his comical relief. Shaheena, Rubina, and Rehmat Kharal, for their advice and concern throughout my research.

I send sincere thanks to dear Anjum Junaid, who included me in every one of her prayers for five years and continues to do so. I thank Sheerose, Muzammil and Saba Kharal for coming to Waterloo for a few weekends just to help me out. I thank Yasmin Muhammad, Kasia Pietroszek, and Irina for their help with the children, and Celeste Loccisano for her many emails of wisdom. I also thank Farzana Arif and Reem Al-Halimi for their reassurance and support.

Lastly and sincerely, I would like to thank God for making this possible.

Contents

1 Introduction **1**

2 DNA Microarray Technology **3**

2.1 Overview 3

2.2 Human Genome 4

2.2.1 The Human Cell 4

2.2.2 Nucleotides 6

2.3 DNA Microarrays 7

2.3.1 Oligonucleotide Arrays 9

2.3.2 Microarray Data Format 10

2.4 DNA Microarray Analysis 11

2.4.1 Dimensionality Reduction of Microarray Data 12

2.5 Applications of Microarrays 13

3 Methods in Manifold Learning **15**

3.1 Introduction 15

3.2 Goals of Dimensionality Reduction 15

3.2.1 Notations and Definitions 16

3.3 Principal Component Analysis 17

3.3.1 Kernel PCA 20

3.4 Linear vs. Nonlinear Dimensionality Reduction 22

3.4.1 Local Linear Embedding 22

3.4.2 Isomap 27

3.4.3 Geodesic Distances 28
3.5 Summary 32

4 Overview of Semidefinite Embedding 33
4.1 Introduction 33
4.1.1 Isometry and Manifold Learning 34
4.1.2 Problem Definition 35
4.1.3 Isometry Constraints 36
4.1.4 Formulation Into A Kernel Matrix 37
4.1.5 Additional Constraints 38
4.1.6 Semidefinite Programming Problem 40
4.1.7 Recovering Outputs 41
4.2 Applications of Semidefinite Embedding 42
4.2.1 Eigenvalue Spectra 43
4.2.2 Synthetic Data 44
4.2.3 Drawbacks 44
4.3 Landmark Semidefinite Embedding 46
4.3.1 lSDE Algorithm 47
4.3.2 Findings: Landmark Semidefinite Embedding 51
4.4 Summary 52

5 KAS Algorithm and Experimental Methodology 53
5.1 Motivation 53
5.1.1 Feature Selection 55
5.1.2 Kernel Alignment 56
5.1.3 Summary of the KAS Algorithm 58
5.2 Experimentation 59
5.2.1 Data Sets 60
5.2.2 Determination of Results 61
5.2.3 Biological Perspective 67

6 Results **69**
6.1 Results: Breast Cancer Data Set 69
6.1.1 Clustering Validation Indices 70
6.1.2 Variance in Two Dimensional Embedding 74
6.2 Results: Colon Cancer Data Set 74
6.2.1 Clustering Validation Indices 76
6.2.2 Variance in Two Dimensional Embedding 78
6.3 Results: Prostate Cancer Data 81
6.3.1 Clustering Validation Indices 82
6.3.2 Variance in Two Dimensional Embedding 85
6.4 Results: AML/ALL Data 85
6.4.1 Clustering Validation Indices 87
6.4.2 Variance in Two Dimensional Embedding 90
6.5 Summary of Results 92
6.5.1 Two Dimensional Output 92
6.5.2 Variance in Two Dimensions 92
6.5.3 Clustering Validation Indices 93

7 Conclusions and Final Remarks **95**
7.1 KAS Summary 95
7.1.1 Inherent Dimensionality 96
7.1.2 Two Dimensional Display 98
7.2 PCA, LLE, Isomap Results 99
7.3 Computational Efficiency 100
7.4 Conclusions and Application 101
7.5 Future Directions 102

A Mathematical Backgound Material **104**
A.1 Definitions 104
A.2 Mathematical Optimization 106
A.2.1 Convex Optimization Problems 107
A.2.2 Semidefinite Programming 109

A.3 Distance Metrics 110
A.3.1 Fractional Distances 110
A.4 Linear Dimensionality Reduction 111
A.4.1 Multidimensional Scaling 111
A.5 Nonlinear Dimensionality Reduction 113
A.5.1 Laplacian Eigenmaps 113
A.5.2 Hessian Eigenmaps 115
A.6 Support Vector Machines 116
A.6.1 Kernel Trick 118

List of Figures

2.1 The cell is the most basic unit of life within the human body. This figure was taken from [14]. 5

2.2 A strand of DNA has a double helix structure. This figure was taken from [39]. 7

2.3 Experimental design used in oligonucleotide arrays. This figure was taken from [22]. 8

2.4 An example of microarray input data. 10

3.1 A sample manifold in three dimensions. This particular manifold is entitled the *Swiss Roll*. (Figure taken from [27]) 23

3.2 An overview of local linear embedding as defined in [65] 24

3.3 Graph of input data where edges exist from each node to its k nearest neighbours. 28

3.4 Computing geodesic distances on a manifold. (Figure taken from [33]) . 29

4.1 (a)Edges only exist from x_i to its neighbours (b)Graph is further connected by adding edges *between* neighbours of x_i. 36

4.2 SDE maps the teapot images taken at 360^o of rotation to a circle where similarly angled images are displayed closer to one another in the circle. This image is taken from [87]. 42

4.3 Spectrum of eigenvalues from SDE outputs on given image data sets. Figure (a) is completely characterized in a single dimension of SDE low dimensional outputs. Figures (e) and (f) indicate that greater than two dimensions are required to fully represent the variance in the high dimensional data sets. However, when comparing to other algorithms, SDE comes closest to correctly estimating the underlying dimensionality of the data sets. This image is taken from [87]. 45

4.4 SDE almost exactly replots the input data, as does HLLE. Isomap and LLE result in some distortion to the original input data. SDE also exactly characterizes the inherent dimensionality of the output data. This is taken from [87]. 45

5.1 A view of gene expression data in a hierarchical clustering. Genes are clustered based on their expression profiles and grouped with like genes in the heirarchy at indicated at the top. 68

5.2 A view of gene expression profiles across 16 clusters. In (a), the profiles for each gene in the cluster is given, and only the mean profiles across each cluster is given in (b). 68

6.1 Two dimensional outputs of dimensionality reduction methods tested on Breast Cancer Data: The output of each algorithm was clustered with K-Means clustering. The circles within each cluster represents the cluster centre. KAS outputs show greater separation between bordering clusters. The number of clusters, r = 6. 71

6.2 Breast Cancer Data: Clustering validation indices for K-Means and GK clustering of two dimensional output data. K-Means clustering is applied in (a) and (b) and GK clustering is applied in (c) and (d). 72

6.3 Comparison of variance in each dimension of output data generated by dimensionality reduction methods. First dimension is dark blue and begins at the bottom of the bar graph. KAS has the highest percentage of variance retained in two dimensions. 75

6.4 Two dimensional outputs for Colon Cancer Data. K-Means clustering is applied to the two dimensional data. The number of clusters, r = 6. . . 77

6.5 Colon Cancer Data: Clustering Validation Indices for K-Means and GK clustering of two dimensional output data of Colon cancer data set.K-Means clustering is applied in (a) and (b) and GK clustering is applied in (c) and (d). 79

6.6 Comparison of Variance in each dimension of output generated by dimensionality reduction algorithms. First dimension is dark blue and begins at the bottom of the bar graph. KAS retains highest percentage of variance in two dimensions, which is close to 80% of total variance. 80

6.7 Prostate Cancer: Clustering validation indices for output data generated by all four dimensionality reduction methods. 83

6.8 Two dimensional outputs for Prostate Data with K-Means clustering applied to the data. Here the number of clusters, r = 8. 84

6.9 Comparison of Variance in each dimension of output generated by dimensionality reduction algorithms. First dimension is dark blue and begins at the bottom of the bar graph. KAS retains highest percentage of variance in two dimensions, which is close to 62% of the total variance contained in the unreduced prostate data set. 86

6.10 Two dimensional outputs for AML/ALL Data. K-Means clustering is applied to the two dimensional data with the number of clusters $r = 10$. 88

6.11 Clustering validation indices for K-Means and GK clustering of the AML/ALL data. 89

6.12 Comparison of Variance in each dimension of output generated by dimensionality reduction algorithms. First dimension is dark blue and begins at the bottom of the bar graph. KAS retains highest percentage of variance in two dimensions, which is close to 68% of the total variance contained in the unreduced AML/ALL data set. 91

A.1 Difference between convex and non convex data sets diagram from (Figure taken from [35]) 108

A.2 Convex and non convex functions. (Figure taken from [35]) 109

A.3 Introductory slides on support vector machines. (Figures modified from [47]) 117
A.4 An example of data that is inseparable by a linear classifier. This figure was taken from [24] 118

Chapter 1

Introduction

DNA microarray technology has lead to a new class of biological experiments where data acquisition of gene activity is possible on a large scale. A typical microarray data matrix contains the expression levels of thousands of genes across different experimental samples. DNA microarray technology has directed the focus of computational biology towards analytical data interpretation. However, when examining microarray data, the size of the data sets and noise contained within the data sets compromises precise qualitative and quantitative analysis. Conventionally, a dimensionality reduction technique may be used to reduce the size of the dataset before further processing. Dimensionality reduction can also provide a low level visual representation of gene behaviour across the samples. A standard objective of microarray data analysis is to better understand the *gene-to-gene* interactions that take place amongst the entire gene pool. However, applications of dimensionality reduction techniques on microarray data have been only partially successful. Dimensionality reduction of microarray data has yet to effectively tackle the problem of finding a low dimensional embedding that provides an accurate visual representation of gene-to-gene interactions. Visual representations of genes in lower dimensions have often been only partially reflective of actual gene-gene relationships indicated by the higher dimensional data set. The level of accuracy in a low dimensional embedding needs improvement for current dimensionality reduction methods that have been applied to microarray data.

In our work, we apply a dimensionality reduction approach that incorporates semidefinite programming and kernel alignment into the dimensionality reduction problem. Our method, KAS *kernel alignment with semidefinite embedding*, uses the semidefinite embedding algorithm first described in [87], and a derivation of kernel alignment [73] in order to successfully tackle the dimensionality reduction of microarray data. *KAS* is a useful visual tool for both the understanding of gene-to-gene interactions and the replacing of a large unmanageable dataset with one of much lower dimensionality. We describe, in detail, the full *KAS* algorithm and how it compares to more conventional dimensionality reduction methods in chapter 5. This thesis is structured as follows: Chapter 2 provides an overview of DNA microarray technology. Chapter 3 is a survey of prior work and discusses three dimensionality reduction methods in use today. Chapter 4 initiates the relevant work for this thesis with a detailed description of the semidefinite embedding algorithm. In chapter 5, we discuss the *KAS* algorithm and explain the experimental methodology applied in this thesis. Results and comparisons are discussed in chapter 6. Conclusions and the future direction of *KAS* are discussed in chapter 7. Appendix A provides additional information on the topics discussed in this thesis and bibliographic references are given in alphabetical order.

Chapter 2

DNA Microarray Technology

2.1 Overview

Since the beginning of this century we have seen the sequencing of the entire human genome in addition to many other organisms such as fruit flies, yeast, and other bacterium [7]. With the ability to recognize and sequence every gene in the human body, the next imperative step in further understanding our genetic make-up is to discover how genes work together in fulfilling functional roles. The field of functional genomics aims to discover the functions and roles of genes in different organisms. DNA microarray technology is a new class of bioinformatics that allows parallel biological data acquisition not possible in the past [61, 25]. Thousands of expression levels of genes can be monitored simultaneously across many different experimental conditions. Therefore, microarray technology plays a vital role in better understanding the functions of genes. Microarray technology challenges the frontiers of data mining and data analysis. As the data from microarray experiments accumulates, it will be crucial to extract biological significance from the data [25]. This chapter provides an overview of DNA microarray technology and the challenges faced by DNA microarray analysis methods. We begin with an introduction to the human genome and some basic genetic concepts.

2.2 Human Genome

A cell is the basic unit of life within the human body. A cell functions by using its genes to produce proteins. Although each cell within an organism will usually contain the same set of genes, there are significant differences in which genes are activated and how they are controlled. The specification of genes is contained on strands of DNA present in the nucleus of the cell. In figure 2.1 we see a detailed depiction of a human cell.

2.2.1 The Human Cell

All cells have a *plasma membrane*, which regulates the movement of any molecules in and out of the cell and protects the cell from dangers in the outside environment. Within the cell, there are many *organelles* such as the *lysosomes* and *mitochondria*. The organelles are small structures that help with the day-to-day operations of the cell. There are important functions performed by each of the organelles within a cell. For the purposes of our introduction to DNA microarrays, we focus on the *nucleus* of the cell. Within the nucleus of the cell lies the *chromatin* and within the chromatin lies the encoding for every gene. The encoding of genes is given by DNA subsections of the chromatin. The specification of each gene requires thousands of tiny units known as nucleotides. Whether any given gene plays a functional role in a particular cell depends on the production of specific proteins associated with each gene. A cell, such as the one given in figure 2.1, does not exist in isolation, but is surrounded by a community of other cells. It is from interactions with other cells, that a cell knows when to begin production of different proteins.

If a protein specified by a gene is to be produced, the DNA sequence is first copied or transcribed into mRNA (*messenger RNA*). After this copy is made, the mRNA transcription is taken to the contents of the cell, in order to produce the required protein. Therefore, the level of mRNA within a cell

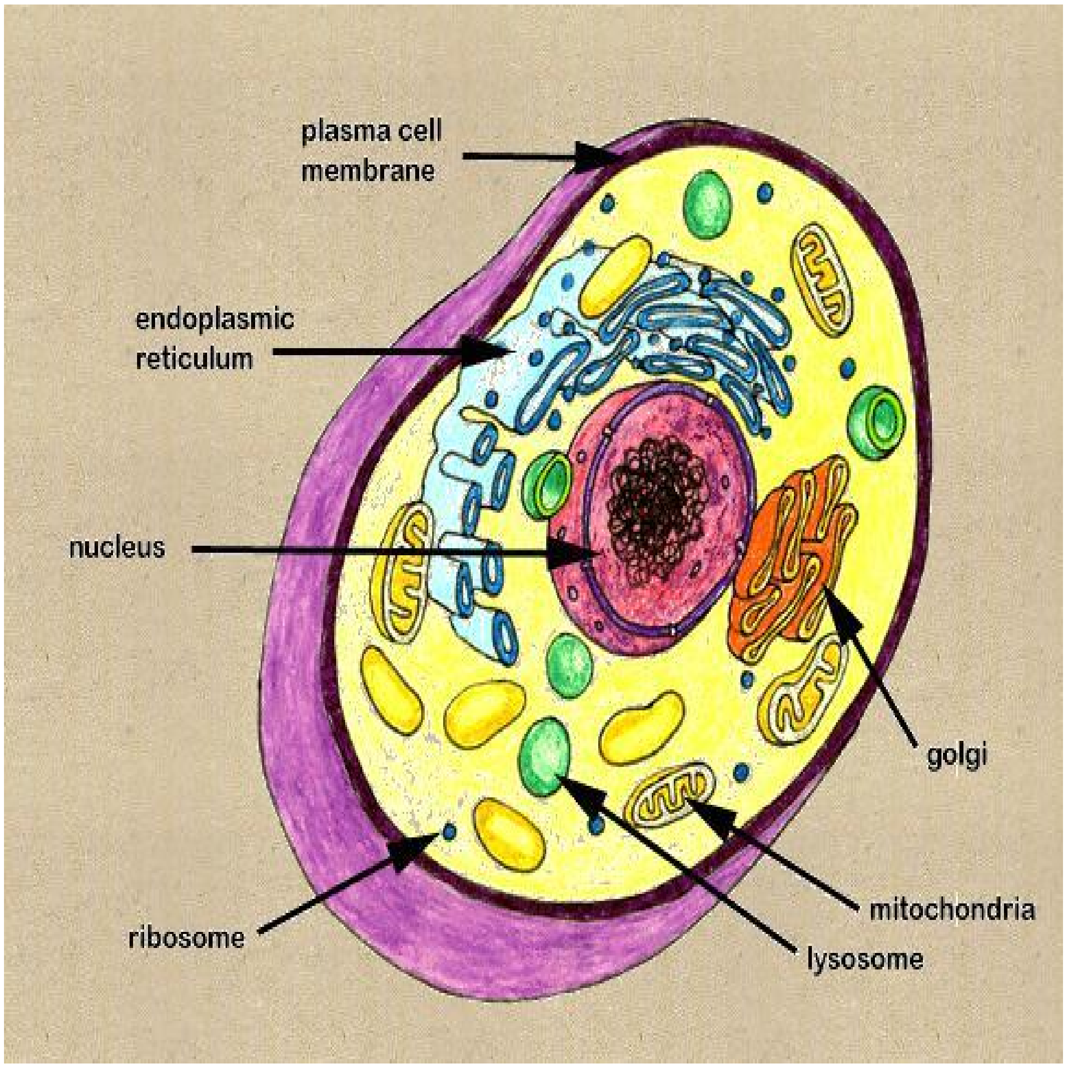


Figure 2.1: The cell is the most basic unit of life within the human body. This figure was taken from [14].

that corresponds to the transcription of a particular gene, is a good indication of whether or not the gene is being used in a given cell at a particular time. Many biological experiments centre around studying the amount of mRNA present in a cell, that corresponds to different genes. In fact, before the advent of DNA microarrays, analysis of this type was at the individual level for one gene at a time. DNA microarrays have allowed the analysis of gene activity within a cell, to occur in parallel for thousands of genes simultaneously. Section 2.3 discusses microarray technology in greater detail. However, in order to better understand the basis of microarray technology we need to first explore some details regarding the copying or transcription of a strand of DNA.

2.2.2 Nucleotides

A nucleotide is the basic unit of DNA. It contains a five-carbon sugar and one of four nucleotide bases (adenine, guanine, cytosine and thymine). Conventionally, each nucleotide is labelled as a letter, one of A,G,C,T. A DNA strand has a double-helix structure as seen in figure 2.2(c). A gene sequence is conventionally given by a string of potentially thousands of nucleotides (i.e. CAGCTCAGGGTTCCCATT...). We see in figure 2.2(b) that the nucleotide bases will pair or *hybridize* with one another through hydrogen bonds and form the double helix displayed in 2.2(c). The nucleotide bases actually hybridize or *bind* exclusively to certain other nucleotides. Adenine will pair with thymine and guanine will pair with cytosine. If the cell has to copy a strand of DNA, the double helix is first unravelled. Next, complementary bases will hybridize with the sequence of bases along the unravelled strand of DNA and form a complementary copy of the original strand of DNA. Thus, when a sequence of DNA is copied we say that it is *transcribed*. Such a process occurs when mRNA is transcribed from a subsequence of DNA in order to produce a protein [14]. The process of transcription plays a crucial role in DNA microarray technology, discussed in the next section.

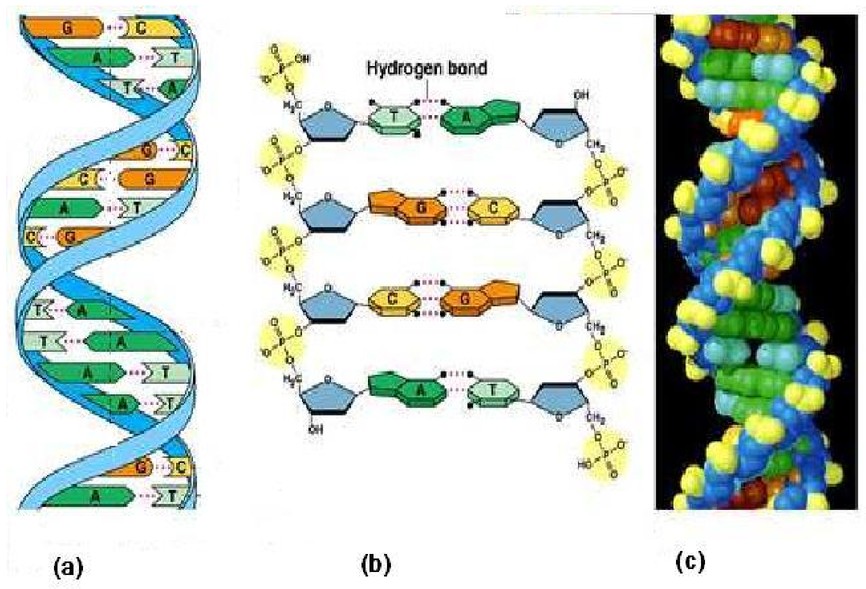


Figure 2.2: A strand of DNA has a double helix structure. This figure was taken from [39].

2.3 DNA Microarrays

In the last decade, molecular biology has seen the rise of a new technology known as DNA microarrays (simplified as *microarrays*). DNA microarrays allow the parallel monitoring of thousands of expression levels of genes simultaneously. The secret behind the technology is the fact that DNA nucleotide bases will hybridize to certain other nucleotide bases. A DNA microarray consists of an orderly arrangement of DNA fragments representing the genes of an organism [15]. Each DNA fragment is a small portion of the nucleotide sequence that represents an entire gene. A microarray consists of up to 200,000 spots on a square glass slide. With the use of microscopic robotics, each spot on the array is set to contain many copies of the nucleotide subsequence that is representative of a single gene. When the DNA microarray is immersed in a cellular sample, the mRNA within the cells will hybridize to complementary strands of DNA contained on the microarray.

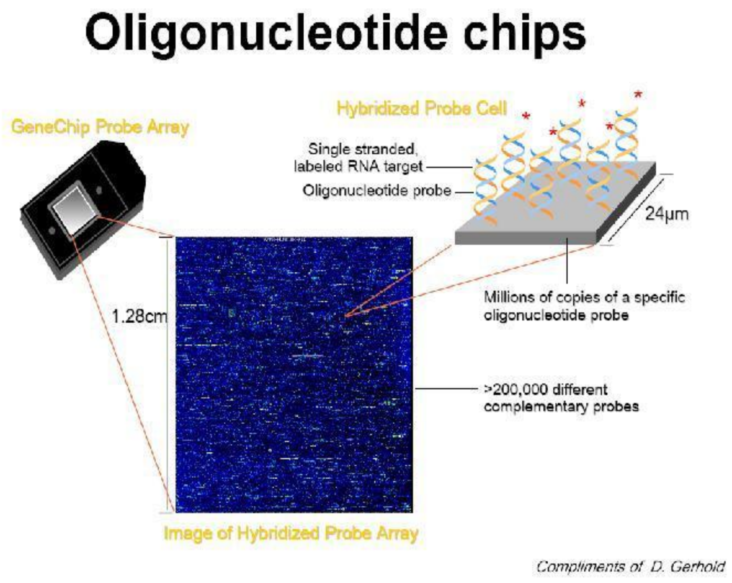

Oligonucleotide chips
GeneChip Probe Array
Hybridized Probe Cell
Single stranded, labeled RNA target
Oligonucleotide probe
24µm
1.28cm
Millions of copies of a specific oligonucleotide probe
>200,000 different complementary probes
Image of Hybridized Probe Array
Compliments of D. Gerhold

Figure 2.3: Experimental design used in oligonucleotide arrays. This figure was taken from [22].

As explained in the next section, fluorescently labelling the mRNA allows us to detect the level of hybridization occuring for a particular gene. The three predominant types of microarray technologies are oligonucleotide arrays, cDNA microarrays, and a related technology known as SAGE (serial analysis of gene expression) [7]. Each technology uses the same principle of measuring the presence of mRNA contained in the cells of a test sample. The aim of any microarray technology is to derive an expression level, as a scalar value, for each gene. High expression levels indicate a high amount of genetic activity for a particular gene, and low expression levels indicate relative inactivity [22]. Throughout this chapter, we will focus on oligonucleotide arrays, and provide a more detailed description of the technology in the next section.

2.3.1 Oligonucleotide Arrays

Oligonucleotide arrays are trademarked as **GeneChip** by the Affymetrix Corporation [16]. These microarrays, often called *chips*, consist of multiple spots or probes of the same DNA fragment. Each DNA fragment is approximately 25 nucleotides in length and is representative of one gene. Therefore, in oligonucleotide arrays the entire gene is not coded. The 25 nucleotide bases are selected such that minimum cross-hybridization occurs with other DNA fragments on the array. Within each probe there are millions of copies of the DNA fragment and there are up to 20 probe pairs for each gene [22]. The purpose of the pairs is discussed in section 2.4. The oligonucleotide array is immersed in a sample containing experimental cells. For example, one sample type may be cancerous colon cells and another sample may be normal colon cells. The mRNA from the sample cells will hybridize to a match of DNA fragments on the chip [15]. The mRNA from the sample is initially extracted and labelled with a fluorescent colour, which allows it to be seen if hybridization occurs. If hybridization does not occur, then the mRNA simply washes off the microarray slide. If a gene is active in a given cellular sample, the mRNA within the cell will hybridize to the DNA subsequence contained on the microarray. This will be demonstrated on the microarray with a fluorescent colour turning high or low. We say that a gene is *expressed* within a sample, if it is transcribed into mRNA and hybridization occurs. Typically a bright colour is used to indicate a relatively abundant expression level and black or no colour indicates that the gene is inactive in the given sample.

An example of the design of oligonucleotide arrays is given in figure 2.3. The amount of hybridization is indicated by the level of fluorescence present in each probe. In order to transform this into a numerical value, the microarray is scanned and each probe is assigned a measure of intensity. The location of the probes and the respective genes that they represent, is crucial in understanding the data generated from microarrays. When the intensity is measured for any given probe, the corresponding gene that it represents must be known. Therefore, following the microarray experiment, each gene is assigned an expression value based on its total expression across all probes [15]. Often multiple experiments are performed, testing similar samples. For example, cancerous colon cells may be extracted from multiple patients. Each patient's sample is used on a separate oligonucleotide array, and the totals are summarized in a matrix. Subsequent tests may be with normal colon cells, tested in a similar manner across many patients without colon cancer. An example of the data format in a matrix is given in figure 2.4.

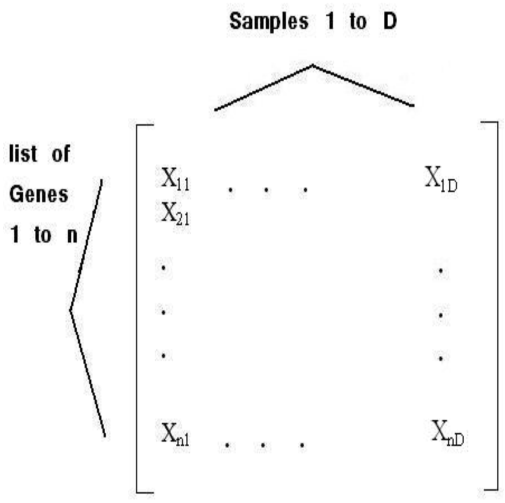


Figure 2.4: An example of microarray input data.

2.3.2 Microarray Data Format

In section 2.3.1 the purpose of microarray technology was discussed. A DNA microarray monitors the expression levels of thousands of genes in the cells of a test sample. Let the number of genes being monitored by an oligonucleotide microarray be represented by n. Therefore, the outcome following one microarray experiment is an expression value for all n genes, as indicated from a particular test sample. Typically, many samples are tested and this results in a matrix or table of expression values. Let the number of test samples be represented by D. Therefore, a DNA microarray, such as an oligonucleotide array, can be represented by expression values stored in a matrix $\mathbf{X}$. The number of rows in $\mathbf{X}$ corresponds to the number of genes, n. The D different samples that the expression values were taken from represent the columns of $\mathbf{X}$. Hence, $\mathbf{X}$ is an $n \times D$ matrix as given in figure 2.4. In the points given below, we introduce the notation used throughout this thesis.

- Let the input data matrix $\mathbf{X}$ contain all n input vectors as the rows of $\mathbf{X}$, where each input vector corresponds to a different gene. Every input vector of $\mathbf{X}$, is represented by x_i where $i \in [1...n]$. The matrix $\mathbf{X}$ is an $n \times D$ matrix. The components of each input vector x_i are represented by x_{ij} where $j \in [1...D]$. The columns of $\mathbf{X}$ represent the different samples tested in the experiment. Therefore, every x_{ij} is the expression level of gene i taken from sample j.

- Matrices are denoted in bold such as $\mathbf{K}$, $\mathbf{M}$ and $\mathbf{G}$. The elements of each matrix are denoted by k_{ij}, m_{ij} and, g_{ij} respectively.

2.4 DNA Microarray Analysis

Once a DNA microarray is ready for analysis, preprocessing steps are typically applied to the raw input data. In theory, the microarray's ability to extract the expression levels of thousands of genes is a tremendous accomplishment. In practice, however, there are many discrepancies or errors that can occur in the data, and this requires the assistance of statistical preprocessing methods. Often there are outliers, or noise within the data that leads to some genes being misrepresented. For example, a common problem when assessing the level of fluorescence present in a probe, is the existence of background noise and non-specific/cross hybridization. The latter term meaning that hybridization or binding occurs between the DNA fragments in a probe and non-target DNA. Background noise can be anything from microscopic chip defects to signal spillage of one probe onto its neighbour within the microarray. Therefore, in order to account for background noise and non-specific binding, the oligonucleotide arrays work with pairs of probes. The pair consists of a perfect match (PM) probe and a mismatch (MM) probe. The mismatch probe is purposely manufactured with an error, whereby the DNA fragments within this probe are different by one nucleotide base. When assessing the level of hybridization for a given probe pair, the hybridization that occurs on the mismatch probe is considered erroneous and is subtracted away from the perfect match probe. This is a common method in dealing with background noise and hybridization errors, however, there are many other approaches in attempting to tackle these problems [7, 22, 15]. Some additional preprocessing steps that are typically applied to microarray data are normalization, taking log_2 of the expression values, scaling the expression values or calculating the expression values based on the PM score subtracting background noise, where some other measure of background noise is calculated [22].

2.4.1 Dimensionality Reduction of Microarray Data

Dimensionality reduction of microarray data involves taking an $n \times D$ input matrix $\mathbf{X}$ and reducing its size while striving to retain much of the information contained within the full data set. This is important because the size of a microarray data set can often reach tens of thousands of genes across potentially hundreds of samples [81]. As DNA microarray technology becomes more and more widespread in the biological community, the need for efficient and accurate analysis methods is crucial. Dimensionality reduction of the data is a critical step in this process. The next chapter extensively discusses the topic of dimensionality reduction, and current methods that are commonly applied to DNA microarrays. Dimensionality reduction of microarray data has two possible approaches; (1) reducing the number of genes or rows in the data while holding the number of columns or samples constant, or (2) reducing the number of columns while keeping the number of rows constant. For the purposes of this thesis, we focus on the latter approach and strive to better understand the relationships between genes in the data set. Dimensionality reduction is often a critical step in the analysis of microarrays, as the computational complexity of analysis methods is growing [57]. Working with a smaller data set, that is still representative of the original, is very desirable. Dimensionality reduction of microarray data also allows one to visually depict the high dimensional data set and subsequently derive meaning. It also assists in a better understanding of the gene-gene relationships that occur in the data set. In this thesis we present a novel approach for the dimensionality reduction of DNA microarray data. This is discussed in chapter 5.

2.5 Applications of Microarrays

DNA microarray technology has broad utilization in the bio-medical community. Measuring the abundance of genetic activity for a particular cell type allows a much deeper understanding of the inner workings of organisms. Some of the immediate ramifications and existing applications of the technology are related to disease discovery [70, 57, 31]. Microarrays allow a comparison between test patient cells and diseased cells for possible early detection of diseases. This type of analysis often requires the use of dimensionality reduction and *support vector machines* (SVM). Support vector machines are briefly introduced in section A.6. This type of analysis alone, has immense benefits and associated functions. Microarrays allow the study of host genomic responses to bacterial infections [22]. Also, there exist applications of microarrays to genotyping, cancer research and measuring mRNA decay rates. A huge benefit of microarray technology exists in the area of drug discovery and design. DNA microarrays have effectively redirected the focus of computational biology. However, as mentioned in section 2.4, DNA microarray technology suffers from a small percentage of genes being misrepresented in the microarray experiments [15]. Therefore, the technology is still definitely in need of improvements. The potential of this technology is largely dependent upon microarray analysis methods and the ability to draw meaning from the data. This thesis focuses on one aspect of microarray analysis, namely dimensionality reduction as discussed in section 2.4.1. In the next chapter we begin with a review of current methods for dimensionality reduction and discuss applications of these methods to microarray data sets.

Chapter 3

Methods in Manifold Learning

3.1 Introduction

DNA microarray analysis, from the onset, has made use of algorithms from a wide range of research areas such as artificial intelligence, statistical learning and data mining [34]. As discussed in chapter 2, due to the nature and size of microarray data, examination of the data calls for the application of dimensionality reduction techniques. In this chapter, we begin with a brief introduction and motivation for dimensionality reduction. Subsequently, three linear and nonlinear dimensionality reduction techniques are examined. A review of the terminology and concepts used in this chapter, are presented in Appendix A.

3.2 Goals of Dimensionality Reduction

In practice, when one acquires data from various sorts of experimentation, the raw format of the data is often high dimensional [48]. Often, to work and perform analysis on such high dimensional data is very difficult and computationally intensive. This type of problem data is often referred to as having *the curse of dimensionality* [48]. Instead, one aims to uncover intrinsic patterns and behaviour that exist in this high dimensional data from a lower dimensional representation of the original data. Searching for such low dimensional structure is broadly known as *dimensionality reduction*. If the original data is perceived as lying on a high dimensional manifold, then *manifold learning* is a way to portray the original data within a lower dimensional manifold. Essentially, a key objective for the dimensionality reduction of high dimensional input data is to obtain a compact representation of the high dimensional data in fewer dimensions [51]. This entails informative visualization of the input space. What is not desired is the loss of valuable information contained within the original manifold. Dimensionality reduction allows for faster analysis and computations on the input data and ideally allows one to focus on only the principal features of the data [84]. The sections that ensue will introduce two categories of dimensionality reduction algorithms. First, one of the principal methods in linear dimensionality reduction is introduced. Subsequently, an introduction to two nonlinear manifold learning algorithms is presented followed by a brief summary. Background material for the algorithms that ensue and additional linear and nonlinear dimensionality reduction methods are covered in Appendix A (sections A.4 and A.5). First, we summarize the notation presented in section 2.3.2, and build upon this in the list given below. Next, we review a set of definitions that are referred to in this chapter.

3.2.1 Notations and Definitions

Notations

- Let the input data matrix $\mathbf{X}$ contain all n input vectors as the rows of $\mathbf{X}$. Every input vector of $\mathbf{X}$, is represented by x_i where $i \in [1...n]$. The matrix $\mathbf{X}$ is an $n \times D$ matrix. The components of each input vector x_i are represented by x_{ij} where $j \in [1...D]$.

- Let the output data matrix $\mathbf{Y}$ contain all n output vectors as the rows of $\mathbf{Y}$. Every output vector $y_i \in R^d$, where $d << D$. Therefore, $\mathbf{Y}$ is an

$n \times d$ matrix. The components of each output vector y_i are represented by y_{ij} where $j \in [1...d]$.

- Matrices are denoted in bold such as $\mathbf{K}$, $\mathbf{M}$ and $\mathbf{G}$. The elements of each matrix are denoted by k_{ij}, m_{ij} and, g_{ij} respectively.

Definitions

- **Manifold**
  A manifold is a topological space which is locally Euclidean [97], meaning that every point has a neighbourhood which resembles Euclidean space and points are separated by Euclidean distances. Examples of manifolds with additional structure include differentiable manifolds and Riemannian manifolds on which distances and angles can be defined [93].

- **Isometry**
  As given in [92], an isometry is a mapping of a metric space onto another or onto itself so that the distance between any two points in the original space is the same as the distance between their images in the second space.

- **Covariance**
  The covariance between x and y, is often written as Cov(x,y). Covariance is the measure of how two variables vary together. The measure of Cov(x,y) becomes more positive for each pair of x and y that differ from their mean in the same direction, and becomes more negative if x and y differ from their mean in opposite directions [89].

3.3 Principal Component Analysis

A common statistical method used for dimensionality reduction of high dimensional data sets is principal component analysis (PCA). PCA is one of the best known unsupervised dimensionality reduction or feature extraction methods [63, 27, 85]. Other methods in the same family as PCA are factor analysis and principal coordinate analysis [59]. The objective of principal component analysis is to find a projection of the input vectors to a low dimensional subspace that maximizes the projected variance in the data [87]. PCA attempts to find a set of d orthogonal vectors that account for as much of the data's variance as possible. This is achieved by calculating the covariance matrix of the inputs $\mathbf{X}$, and subsequently finding the set of eigenvectors and eigenvalues of the covariance matrix. The eigenvectors of the covariance matrix are maintained in descending order based on the corresponding eigenvalues. The result of this eigendecompostion of the covariance matrix is a d dimensional embedding of the original D dimensional data. The eigenvectors are the principal components of the data in d dimensions and define the embedding coordinates in the outputs $\mathbf{Y}$. The eigenvectors are the columns of the output matrix $\mathbf{Y}$. The eigenvalue that corresponds to each eigenvector is a measure of the total variance maintained in this particular eigenvector. Therefore, the d dimensions of the embedding space are defined by the number of eigenvalues having a value that is sufficiently larger than zero. We refer to this as the number of nontrivial eigenvalues. There is a one-to-one correspondence between the input vectors and output vectors, obtained from the eigenvectors of the covariance matrix; that is every output vector y_i is a low dimensional representative of the original input vector x_i. In what follows, an outline of the PCA algorithm from [72, 43] is summarized.

1. Given a random input population $\mathbf{X}$, where $\mathbf{X}$ is an $n \times D$ matrix, define the mean of each column vector to be μ_j for $j \in [1...D]$. We centre the input matrix $\mathbf{X}$ by subtracting the column means from each element x_{ij}:

$$x_{ij} \leftarrow (x_{ij} - u_j). \tag{3.1}$$

This results in the row vectors of $\mathbf{X}$ having a sum of zero:

$$\sum_{i=1}^{n} x_i = 0. \tag{3.2}$$

2. Define a covariance matrix $\mathbf{C}$, given in equation (3.3):

$$\mathbf{C} = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^t. \tag{3.3}$$

3. The covariance matrix $\mathbf{C}$ is an $(n \times n)$ positive definite matrix, having nonnegative eigenvalues. Therefore, when performing eigenanalysis, the following eigenvalue problem is solved:

$$\lambda v = \mathbf{C} v. \tag{3.4}$$

4. To create an ordered orthogonal basis, simply order the eigenvectors in descending order based on their respective eigenvalues. The first eigenvector will have the largest proportion of variance of the original data. The outputs $\mathbf{Y}$ are contained in the highest d eigenvectors of $\mathbf{C}$ having nontrivial corresponding eigenvalues.

5. Let the d eigenvectors of $\mathbf{C}$ represent row vectors in a matrix $\mathbf{V}$. The d dimensional output vectors y_i are derived from the matrix $\mathbf{V}$ and centred inputs x_i, as defined in equation(3.5) [36].

$$y_i = \mathbf{V} x_i^T. \tag{3.5}$$

Ideally when PCA is applied to an input matrix $\mathbf{X}$, for purposes of dimensionality reduction, one would aspire to have the principle variance of the data contained in the top two or three eigenvectors. This allows an embedding into two or three dimensions. However, this would require that only the first two to three eigenvectors contain the principal variance of the data, which is not always the case. In order to assess the inherent dimensionality of the data, we examine the number of nontrival eigenvalues of the covariance matrix $\mathbf{C}$.

Principal component analysis and related schemes, such as factor analysis [69] and singular value decomposition [52, 68], have had much popularity with gene expression data [59]. When any dimensionality reduction method such as PCA is applied to DNA microarray data [61, 60, 85], either the genes (rows in $\mathbf{X}$) or the experimental conditions (columns in $\mathbf{X}$) can be the variables. A significant drawback to the PCA algorithm is that, by itself, PCA does not generalize to nonlinear dimensionality reduction. Nor does it provide a prediction function that can be applied to new inputs. It is this fact, in part, that motivated extensions to the original PCA algorithm. Kernel PCA is one such extension to the original PCA algorithm and is the subject of the next section.

3.3.1 Kernel PCA

Given the success of PCA with feature extraction of linear data, kernel PCA was meant to improve the quality of the PCA algorithm on nonlinear data. Assuming there is a body of problems that exists having data with nonlinear relationships, can one apply linear analysis methods and still draw significant features from the data? This question is the main focus behind kernel PCA. The basic idea is to project the original data into an even higher dimensional space, called a feature space, and work with linear methods within such a feature space. However, this methodology of projecting data into higher dimensions can result in an infeasible problem due to the computational burdens of explicitly calculating the projection function. Instead, the *kernel trick* [43] is employed whereby one no longer needs to explicitly calculate the projection function. Kernels and the kernel trick is discussed in section A.6.1. The kernel approach in equation (3.6) allows computations to work with dot products of the projection function, Φ, and adapts the covariance matrix of the traditional PCA algorithm above to work with the kernel function. Each element c_{ij} of the covariance matrix $\mathbf{C}$ is defined in equation (3.7).

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle \,. \tag{3.6}$$

$$c_{ij} = \sum_{i=1}^{n} \Phi(x_i)\Phi(x_j)^t. \tag{3.7}$$

Applying the derivation given in [57, 24], kernel PCA reduces to an eigenvalue problem of the kernel matrix $\mathbf{K}$ as follows:

$$\lambda v = \mathbf{K} v. \tag{3.8}$$

In the same manner as PCA, we solve the eigenvalue problem given in equation (3.8) and derive the d dimensional embedding of the original inputs. However, a difference in the kernel PCA method is the ability to derive a prediction function for new inputs [24]. Let v_{kq} represent the q^{th} coordinate of the k^{th} eigenvector of $\mathbf{C}$. Given λ_k, the k^{th} eigenvalue of $\mathbf{C}$, we define $l_k = \lambda_k n$. Therefore, an output projection function P_k, for a test point x_t, is given in equation (3.9). The full derivation for equation (3.9) can be seen in [4]. For example, if a two dimensional prediction for a test point is desired, we derive $P_1(x_t)$ and $P_2(x_t)$ based on the two eigenvectors that contain the highest percentage of variance of the original data.

$$P_k(x_t) = \frac{1}{\sqrt{l_k}} \sum_{q=1}^{n} v_{kq} K(x_t, x_q). \tag{3.9}$$

We see that the difference between PCA and kernel PCA is to a great extent this predictive function for new test points, and the fact that kernel PCA is more suited to nonlinear input data. Similar to PCA, kernel PCA has been evaluated on microarray data sets. In [57] Pochet *et al.* evaluate kernel PCA with different kernel functions to assess the role of nonlinearity in dimensionality reduction of DNA microarray data. In recent years alternative nonlinear algorithms based on PCA, such as total principal component regression (TPCR) in [76], have been applied to microarray data with positive results showing improvement over traditional PCA. An alternative widely held method in dimensionality reduction of microarray data is multidimensional scaling. Section A.4 in Appendix A discusses multidimensional scaling at greater length.

In the past, linear dimensionality reduction methods such as PCA and MDS were popular analysis tools for microarray data. However, recent years have seen the rise of newer dimensionality reduction or manifold learning methods.

These methods are more sensitive to nonlinear input data. We proceed with a discussion of nonlinear dimensionality reduction.

3.4 Linear vs. Nonlinear Dimensionality Reduction

The need for nonlinear dimensionality reduction techniques is motivated by the fact that linear methods such as PCA and MDS have difficulty in detecting the intrinsic dimensionality of high dimensional data, and fail to learn complicated nonlinear structure that may exist in the data [10]. In the past five to six years, a handful of nonlinear dimensionality reduction algorithms have been proposed from manifold learning theory. Common to all methods, including PCA and MDS, is the application of eigendecomposition to a matrix $\mathbf{M}$, where $\mathbf{M}$ is representative of the original problem data set in a well defined manner. As in PCA and MDS, the eigenvectors having the largest nontrivial eigenvalues provide a low dimensional embedding of the originally high dimensional data set. We begin with the *local linear embedding* algorithm followed by an overview of the *Isomap* algorithm. Appendix A provides a summary of two additional nonlinear dimensionality reduction methods, namely *Hessian* and *Laplacian Eigenmaps*.

3.4.1 Local Linear Embedding

The locally linear embedding (LLE) algorithm of Roweis and Saul [66] tries to find the best d dimensional projection of the data which preserves global nonlinear geometry. We see an example of a nonlinear manifold in figure 3.1. The LLE algorithm will take such nonlinear input data and attempt to maintain local patches from the original manifold to the embedding space. LLE assumes that if there is sufficient sampling from a high dimensional manifold, each point will lie on a locally linear patch with its neighbours. Every input vector x_i is approximated by a weighted linear combination of its neighbours.

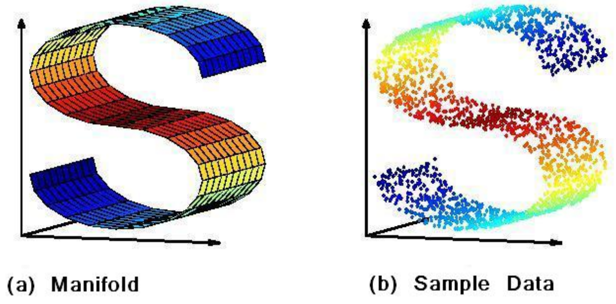


Figure 3.1: A sample manifold in three dimensions. This particular manifold is entitled the *Swiss Roll.* (Figure taken from [27])

.

These weights are subsequently used to define a similar neighbourhood in $d << D$ dimensions. A brief, three step overview of the LLE algorithm [65] is given in figure 3.2. A more detailed description of the LLE algorithm given in [27, 64] follows:

1. Given n input vectors in D dimensions, find all k nearest neighbours for each input vector x_i. Let the neighbours of x_i be represented by the matrix N_i

2. For every input x_i, construct a weight matrix $\mathbf{W}$ from N_i. Each element w_{ij} of $\mathbf{W}$ represents the weight coefficient between input x_i and its neighbour x_j. The construction of $\mathbf{W}$ must hold to the constraint given in step (3).

3. Reconstruct x_i as $\hat{x}_i$ from the neighbours of x_i, and their respective weights w_{ij}. We see that $\hat{x}_i$ is defined as:

$$\hat{x}_i = \sum_{j=1}^{n} w_{ij} x_j. \tag{3.10}$$

   The matrix of weights $\mathbf{W}$ must be chosen such that the error, *E(W)*, is minimized between x_i and $\hat{x}_i$. $\mathbf{W}$ is defined by equation (3.11), and

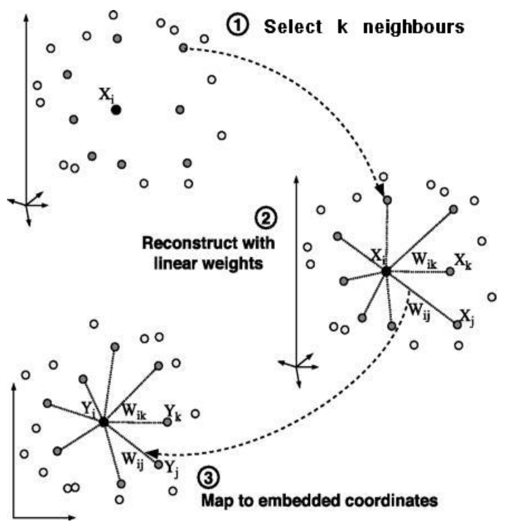


Figure 3.2: An overview of local linear embedding as defined in [65]

.

constraints (a) and (b) below, in order to preserve local isometry of the input vectors in a low dimensional embedding.

$$E(W) = \sum_{i=1}^{n} |x_i - \sum_{j=1}^{n} w_{ij} x_j|^2. \tag{3.11}$$

In minimizing the cost function in equation (3.11), we restrict the matrix of weights **W** with the following constraints:

(a) For all x_i and x_j that are *not* neighbours, $w_{ij} = 0$.

(b) The row sums of **W** must be equal to 1. This is defined in equation (3.12). Essentially, the constraint in equation (3.12) will allow the reconstruction weights to be invariant to translations for any particular input vector and its neighbours [64].

$$For\ all\ i \sum_{j=1}^{n} w_{ij} = 1,\ for\ all\ i. \tag{3.12}$$

4. The matrix of weights **W** is used to reconstruct the output vectors y_i, as given in equation (3.13). Let $\hat{y}_i$ represent the reconstructed output vectors for all $i \in [1...n]$.

$$\hat{y}_i = \sum_{j=1}^{n} w_{ij} y_j. \tag{3.13}$$

5. The final embedding coordinates **Y** can be obtained by minimizing the cost function in equation (3.14). In equation (3.11) we optimize the matrix of weights **W**, given that the input vectors x_i are held constant. The optimization in equation (3.14) optimizes the output vectors y_i, given that the weight matrix is held constant. The output vectors y_i are also restricted to be centred about the origin in equation (3.15). This is in order to remove a translational degree of freedom from equation (3.14).

$$\phi(Y) = \sum_{i=1}^{n} |y_i - \sum_{j=1}^{n} w_{ij} y_j|^2. \tag{3.14}$$

$$\sum_{i=1}^{n} y_i = 0. \tag{3.15}$$

6. Construct a sparse Hermitian matrix **M** representing the matrix of weights **W** Appendix A, section A.1 discusses Hermitian matrices). Calculate low dimensional projection of the original inputs by performing eigendecomposition on the matrix **M** as given in equation (3.16).

$$\lambda v = \mathbf{M} v. \tag{3.16}$$

   The output vectors in **Y**, that give the final d dimensional embedding coordinates, are directly equal to a subset of the eigenvectors generated by equation (3.16). In this case, the selected eigenvectors are the first *d+1* eigenvectors derived from the eigendecomposition of **M**. The matrix **M** is an extremely sparse matrix [27].

Due to the nature of the LLE algorithm, particularly constraint (b) from step 3, the first eigenvector of **M** is always a vector of ones. Its corresponding eigenvalue is also equal to one. This eigenvector and eigenvalue are essentially ignored when deriving the final embedding coordinates. Discarding this eigenvector is necessary in order to ensure that the outputs sum to zero (equation (3.15)). The components of the remaining eigenvectors sum to zero and the constraint in equation (3.15) is satisfied by virtue of orthogonality [27]. Therefore, the d dimensional embedding is obtained by selecting the first $d+1$ eigenvectors of $\mathbf{M}$. We observed that the final LLE problem is simplified into the *sparse* eigendecomposition problem given in equation (3.16). This gives the LLE algorithm an advantage of simplicity over other nonlinear dimensionality reduction methods such as Isomap (discussed in the next section). A key difference behind the LLE algorithm when compared to other predecessor dimensionality reduction methods is the concept of isometry (discussed in section 3.2.1). By formulating a mapping of x_i from its neighbours and associated weights, the projection of x_i is simply a rotation plus translation to a lower dimensional embedding. LLE is computationally simple and unsupervised, thus the algorithm has gained much interest since its formulation in 2000 [66]. A drawback of the algorithm is the fact that the eigenvalue spectra does not provide significant insight into the inherent dimensionality of the high dimensional data set [87]. This was due to the fact that the largest eigenvector derived from equation (3.16) is a vector of ones and the smallest eigenvector is always a vector of zeros. Therefore, the eigenvalue spectra of the final embedding vectors is not representative of the overall variance retained from the high dimensional data set and is derived based on the contraint in equation (3.12) of the LLE algorithm.

The effectiveness of LLE on DNA microarray data was reviewed in [70], where Chao and Lihui applied LLE to data sets dealing with cancer. Chao and Lihui also proposed a revised form of LLE using fractional distance metrics and compared this to the original LLE algorithm (see section A.3.1 for an overview of fractional distance metrics). Some improvements were seen by employing fractional distance metrics to select the k nearest neighbours.

In their experiments the accuracy of LLE on microarray input data was evaluated by the classification accuracy of a support vector machine. It was seen that accuracy results with LLE (Euclidean or fractional metrics) were consistently better than those of prior dimensionality reduction approaches such as PCA, correlation coefficient (CC) and signal to noise ratio (SNR). In [28] a novel clustering approach called fuzzy map clustering (FMC) was proposed for clustering microarray data. FMC linearly approximates the fuzzy memberships of neighbouring objects in a manner that is similar to steps 2 and 3 of the LLE algorithm. FMC runs in reasonable execution time similar to LLE, and yields informative clusters of microarray data [28]. In their work in [42], Katagiri and Glazebrook introduced a new nonlinear dimensionality reduction approach for pattern recognition given the name local context finder (LCF). LCF was also adapted from the original LLE algorithm by Roweis and Saul [66], and was found to be a robust method for extracting information from microarray data. As mentioned above, the LLE algorithm as formulated in [66], does not easily allow analysis of the eigenvalue spectra of the matrix $\mathbf{M}$. Therefore, it is difficult to assess the inherent dimensionality of the input data set. We have seen that LLE uses the notion of a weight function assigned to each neighbour in order to find a low dimensional approximation to the input data. Isomap employs a different mechanism when assessing neighbourhoods in the high dimensional space, which is the topic of discussion in the next section.

3.4.2 Isomap

The underlying motivation for the Isomap algorithm, given in [77], is similar to that of LLE. The problem is as follows: Given a Riemannian manifold in high dimensional space, find a lower dimensional submanifold that maintains local distances. Whenever the idea of local distances arises in manifold learning, one immediately thinks of a neighbourhood. Isomap is short for *isometric feature mapping* [27]. In order to reveal an isometric feature mapping, the algorithm takes into account geodesic distances between neighbouring

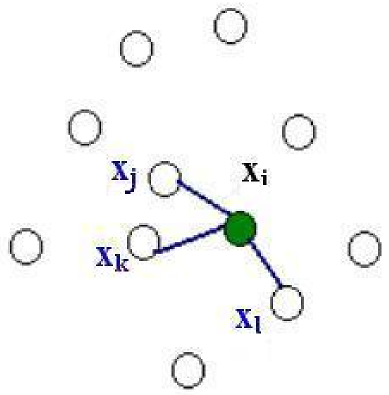


Figure 3.3: Graph of input data where edges exist from each node to its k nearest neighbours.

points on the manifold. Geodesic distances are discussed in the next section. In the Isomap algorithm, a distance matrix $\mathbf{M}$ is computed using geodesic distances between input vectors. The geodesic distances are typically approximated using a shortest paths algorithm. Subsequently, MDS is applied to the covariance matrix of M, from which the low dimensional embedding is extracted in a routine manner using eigenanalysis [27]. We first provide an overview of geodesic distances, followed by the rudiments of the Isomap algorithm.

3.4.3 Geodesic Distances

In manifold learning, we are often interested in the shortest path between points on the manifold in high dimensional space, and approximating this distance in a representative manifold embedded in lower dimensional space. Typically, Euclidean distance is used as an approximation of the shortest path between points on the high dimensional manifold. One can imagine as the dimensionality of the dataset increases, Euclidean distance may not

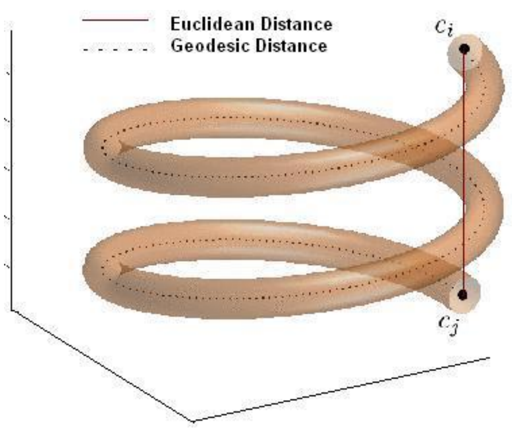


Figure 3.4: Computing geodesic distances on a manifold. (Figure taken from [33])

be an accurate representation of true shortest path between points within the framework of the manifold. This was discovered by Aggarwal *et al.* [2] and discussed in section A.3.1. For example in figure 3.4, the shortest path between c_i and c_j is a line straight across the manifold. However, if one was to maintain the restriction that the manifold must be traversed in determining the shortest path, then the shortest distance between c_i and c_j would be much larger. This is given as the *geodesic* distance between the points. As mentioned in [77, 27, 91] the geodesic distance is believed to be a fundamental distance metric when considering a manifold's structure. The Isomap algorithm employs a geodesic approximation within the algorithm whereby geodesic distances are approximated by the shortest path algorithm.

The key steps in the Isomap algorithm are; (1) build a graph of all input vectors that is locally connected to the k nearest neighbours (2) measure pairwise distances by the length of the shortest paths to each neighbour. This length is an approximation to the geodesic distance between a node and its neighbours (3) Lastly, MDS is used to find a set of low-dimensional points with similar pairwise distances [27, 77]. A more detailed description follows:

1. Find k nearest neighbours of every input vector x_i using standard Euclidean metrics (another approach is to choose all neighbours within a radius e).

2. Construct an undirected graph G. Let all inputs x_i be represented by nodes in G with neighbours of x_i being connected by edges. An example of such a graph can be seen in figure 3.3. For example, the three neighbours of x_i are labelled x_j , x_k and x_l.

3. For every connected component of G, compute the shortest distances between all neighbours, such as x_j and x_k: this is taken to be the sum of edges along the shortest path between the two nodes. The shortest path algorithm is an approximation to geodesic distances [62, 23].

4. Construct a matrix $\mathbf{M}$ of all shortest paths between nodes. Compute a covariance matrix $\mathbf{C}$ of the matrix $\mathbf{M}$. Perform MDS of the covariance matrix $\mathbf{C}$, and thereby find a low dimensional embedding representative of the original high dimensional inputs. The final embedding coordinates are derived from the final eigenvectors and eigenvalues of $\mathbf{C}$. This is similar to equation (A.11) of MDS.

Given a *well sampled* manifold in high dimensional space, geodesic distances can be accurately approximated using shortest path algorithms as given in step 3. Proof for this can be seen in [77, 10]. The Isomap algorithm, as tested by Tenenbaum *et al.* [77], is able to recover an underlying Euclidean manifold reasonably well, given that the underlying manifold intrinsically represents a convex region of Euclidean space [27]. The Isomap algorithm, incorporating geodesic distance calculations discussed in step 3, is tested on microarray data in [40]. In their experiments, Nilsson *et al.* compared the Isomap algorithm to standard Euclidean metrics. Their findings indicate that the Isomap algorithm was much more accurate in the visualization of relationships between genes. Their conclusions highlight the relevance and

applicability of nonlinear dimensionality reduction methods to microarray data sets, and suggest that the Isomap algorithm is a reliable tool for the analysis and interpretation of microarray data [40]. Lu and Wu [44] in their work applied an innovative version of the Isomap algorithm to the microarray data set from Spellman *et al.* [58]. They employ the Isomap algorithm followed by a wavelet transformation. The algorithm is termed, *Isomap + DWT* (nonlinear dimensionality reduction and wavelet transform). The findings in [44] show that *Isomap + DWT* produced an improved visualization of data over traditional Isomap (see figure 1 of [44]). Zhang and Zhang, authors of the GMap software tool [96], examined the effectiveness of Isomap for analysis of public microarray data. They also proposed slight modifications to the original algorithm in [77], offering additional distance metrics for choosing the initial k nearest neighbours, and proposed a method to ensure the connectivity of the nearest neighbour graph.
The original Isomap algorithm in [77] does not simplify to a sparse eigenvalue problem and is quite computationally expensive, $O(n^3)$, where n is the number of inputs [78]. It was noted in [21] that the Isomap algorithm is primarily suited to data sets having an underlying convex subset from which the low dimensional embedding is extracted. Tenenbaum *et al.* applied the idea of landmarks to the Isomap algorithm to improve the computational overhead [20]. Initially, a set of $l << n$ landmark vectors are chosen from the set of all input vectors in $\mathbf{X}$. Isomap was first applied to only the landmark vectors. Based on the embedding coordinates of the landmarks, the embedding coordinates of the remaining n-l input vectors were estimated. Landmark Isomap was seen to improve the complexity costs to $O(l^2n)$. However this comes at the price of somewhat less accuracy in visualizing the input data. A detailed analysis of the landmark approach with Isomap can be seen in [20].

3.5 Summary

In this chapter, we have seen both linear and nonlinear dimensionality reduction methods applied to DNA microarray data. The literature indicates that there are improvements achieved in the application of nonlinear dimensionality reduction techniques, over traditional linear methods such as PCA and MDS. It was noted in section 3.4.1 and 3.4.2 that nonlinear algorithms such as LLE and Isomap, having successful application to microarray data, are limited in some respects. For example, Isomap assumes that the underlying low dimensional manifold is convex which is not always the case [21]. However, the Isomap algorithm does provide the eigenvalue spectra of the underlying manifold, and therefore allows one to determine the intrinsic dimensionality of the data set. LLE does not allow this type of analysis, as the eigenvalue spectra is not clearly indicative of such a relationship. Given that the area of nonlinear dimensionality reduction is relatively new [66, 77, 21, 27], there is room for new and improved nonlinear dimensionality reduction or manifold learning techniques. The LLE and Isomap algorithms perform dimensionality reduction of a high dimensional data set, based on the manner in which a neighbourhood is defined. Each algorithm employs a different mechanism in characterizing a neighbourhood in high dimensional data, and uses this to find a low dimensional embedding which maintains the neighbourhood properties. The chapter that ensues will cover a new method in nonlinear manifold learning which defines a more restrictive measure for a neighbourhood. The algorithm, semidefinite embedding, is discussed in chapter 4.

Chapter 4

Overview of Semidefinite Embedding

4.1 Introduction

In section 3.3 and A.4.1 the PCA and MDS algorithms were discussed. These two linear dimensionality reduction techniques can be classified as linear spectral embedding of high dimensional data. Linear spectral embedding methods perform eigenanalysis on a matrix $\mathbf{M}$ that is a *linear* representative of the original input vectors in $\mathbf{X}$. Other methods that were discussed in chapter 3, such as Isomap, apply eigenanalysis to *nonlinear* representations of the original data matrix $\mathbf{M}$. This is classified as nonlinear spectral embedding. Semidefinite embedding (SDE), as introduced by Weinberger and Saul [87], is another form of nonlinear spectral embedding. Eigenanalysis of a representative matrix $\mathbf{M}$ is performed and the highest order d eigenvectors, containing the principle variance of the data, are extracted to view a low dimensional portrayal the data. SDE derives a low dimensional manifold by employing techniques from semidefinite programming. This allows the definition of constraints and tighter control on the learned outputs $\mathbf{Y}$. In this chapter we will discuss the semidefinite embedding algorithm and its descendant, landmark semidefinite embedding.

4.1.1 Isometry and Manifold Learning

Semidefinite embedding, similar to other manifold learning methods, entails the detection of a low dimensional manifold from a set of data points drawn from the original high dimensional manifold. Semidefinite embedding aims to discover such a manifold with the property that it is locally isometric to the original. As discussed in section A.1, an isometry is a mapping of a metric space onto another such that distances between points are maintained in the mapping [92]. For example, in a three dimensional plane any combination of a rotation plus a translation is an isometry of the plane. Thus, the problem in semidefinite embedding, as characterized by Weinberger and Saul [87], requires finding a smooth and invertible mapping that preserves local distances and behaves locally like a rotation plus a translation. In this manner SDE is similar to algorithms such as LLE and Isomap. In their work, Weinberger and Saul [87] apply the concept of isometry to data sets. They find a d dimensional isometric embedding for the original D dimensional input space. The SDE, LLE and Isomap algorithms primarily differ in their respective definitions of a neighbourhood. We saw in section 3.4.1 that the LLE algorithm calculates the best coefficients to approximate each input vector on the high dimensional manifold, by a weighted linear combination of its neighbours. Subsequently, LLE tries to find a set of low-dimensional output vectors, which can be linearly approximated by its neighbours using the same coefficients. It was also seen in section 3.4.2 that the Isomap algorithm defines a neighbourhood based on geodesic approximations. Geodesic distances between input vectors in the high dimensional data are approximated using a shortest path algorithm. This distance between neighbouring points is again used as a basis to derive a low dimensional embedding. The semidefinite embedding algorithm also tries to perserve neighbourhood relationships from the high dimensional manifold to a low dimensional representation. However, as we shall see, the SDE algorithm attempts to tighten the constraints on what defines a neighbourhood. We now go step by step through the mechanics of the semidefinite embedding algorithm. The notation that we use in this thesis was introduced in section 3.2.1, and briefly reviewed in section 4.1.2.

4.1.2 Problem Definition

Consider a D dimensional input space and n input vectors x_i, where i $\in$ $[1...n]$. Therefore, all input vectors $x_i \in R^D$. Let y_i represent n output vectors of dimension d, where $d << D$ and i $\in [1...n]$. It follows that all output vectors $y_i \in R^d$. Define a neighbourhood matrix $\mathbf{A} \in R^{n \times n}$, where all $a_{ij} \in [0, 1]$, and each entry is indicative of a neighbourhood relationship between two input vectors. If $a_{ij} = 1$, this indicates the inputs i and j are neighbours and accordingly, if $a_{ij} = 0$, this indicates they are *not* neighbours. Let $\mathbf{A^x}$ represent a neighbourhood matrix defined for $\mathbf{X}$ and let $\mathbf{A^y}$ represent a neighbourhood matrix defined for $\mathbf{Y}$. It follows that each entry, a^x_{ij} in $\mathbf{A^x}$, is indicative of a neighbourhood relationship between x_i and x_j across all $i, j \in [1...n]$. This holds similarly for $\mathbf{A^y}$. Therefore, $\mathbf{Y}$ is a representative locally isometric data set of $\mathbf{X}$, if for every a^x_{ij}=1 in $\mathbf{X}$, there exists a^y_{ij}=1 in $\mathbf{Y}$. That is, for every neighbourhood in $\mathbf{X}$, there exists a rotation plus translation to a similar neighbourhood in $\mathbf{Y}$. Given that the semidefinite embedding algorithm employs a semidefinite programming methodology, a set of constraints must be applied to the semidefinite program that is solved. A semidefinite program seeks to optimize the value of an unknown, given a set of constraints. See Appendix A, section A.2 for an introductioni to semidefinite programming and convex optimization theory. In the case of SDE, the optimization variable will be learning the outputs $\mathbf{Y}$. The set of constraints comes from the isometric restrictions placed on the outputs. Thus, the problem definition leads Weinberger and Saul [87] to a set of equality constraints on the learning of $\mathbf{Y}$.

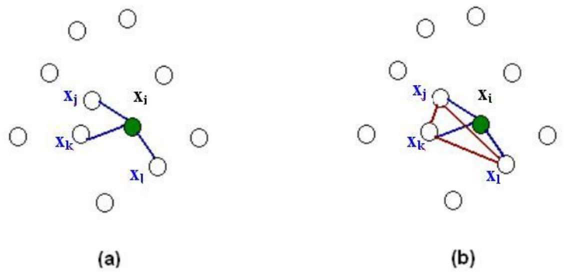


Figure 4.1: (a)Edges only exist from x_i to its neighbours (b)Graph is further connected by adding edges *between* neighbours of x_i.

4.1.3 Isometry Constraints

In order to preserve isometry in a low dimensional embedding of the high dimensional input space, it was explained in section 4.1.2 that neighbourhoods from the input space $\mathbf{X}$ must be preserved in the output space $\mathbf{Y}$. Let G represent a graph of all input vectors as given in figure 4.1(a). Let us examine one input vector x_i and its neighbours x_j, x_k and x_l. The distance to each neighbour is represented by the length of the connecting edge. We see that local isometry will only exist if edge lengths representing distances to neighbours, and the corresponding angles of those edges are preserved from neighbourhoods in $\mathbf{X}$ to neighbourhoods in $\mathbf{Y}$. Thus, for every x_j and x_k that are neighbours of x_i, the dot products of the edge lengths to each neighbour must be preserved. This not only preserves the distance to each neighbour of x_i, but also preserves the angles. This is defined in equation (4.1).

$$\langle (y_i - y_j), (y_i - y_k) \rangle = \langle (x_i - x_j), (x_i - x_k) \rangle \,. \tag{4.1}$$

Next, Weinberger and Saul [87] take the idea of neighbourhoods one step further. Examine figure 4.1(b). We see that figure 4.1(a) is simply a graph of all neighbourhood relationships on x_i. Here the number of neighbours, k, is equal to three. In figure 4.1(b), the graph is further connected by adding edges between x_j, x_k and x_l. The further connected neighbourhood in figure 4.1(b) indicates that x_j, x_k and x_l are also neighbours of each other, leading to the formulation of triangles between x_i and its neighbouring pairs. Weinberger and Saul [87] apply the further connectivity constraint in figure 4.1(b) to the semidefinite embedding algorithm. This ensures that tight connectivity constraints exist on all neighbourhoods, and allows the constraint in equation (4.1) to be reformulated as a difference of square distances. The reformulation of equation (4.1) into a difference of square distances, equation (4.2), is due to an inherent property of triangles. When lengths of all edges of a triangle are preserved, the corresponding angles between edges are also preserved.

$$\|y_i - y_j\|^2 = \|x_i - x_j\|^2 \text{ , } \textit{for all i,j.} \tag{4.2}$$

It is clear that equation (4.2) applies to all edges between x_i and its neighbours x_j, x_k and x_l (from figure 4.1). However, due to the additional edges between x_j, x_k and x_l, equation (4.2) will apply to the edge between x_k and x_l, x_k and x_j, and x_j and x_l. Thus, the lengths of all sides of the three triangles in figure 4.1(b) are preserved by the constraint in equation (4.2) and therefore, the angles to each neighbour are preserved. The problem of learning the output vectors in $\mathbf{Y}$ for a locally isometric manifold in $d << D$ dimensions, now has the notion of preserving local isometry given in equation (4.2).

4.1.4 Formulation Into A Kernel Matrix

In section 4.1.3 we derived the contraints necessary for the preservation of local isometry in the low dimensional embedding. Next, in order to correctly formulate the semidefinite embedding algorithm as a semidefinite program, Weinberger and Saul redefine the output vectors in $\mathbf{Y}$ in terms of a kernel matrix $\mathbf{K}$. The problem becomes learning the kernel matrix $\mathbf{K}$, and subsequently recovering the output vectors for the low dimensional embedding. The kernel matrix $\mathbf{K}$ is defined as the dot product between output vectors y_i, where i $\in [1...n]$. $\mathbf{K}$ is an $n \times n$ symmetric matrix:

$$k_{ij} = \langle y_i, y_j \rangle \, . \tag{4.3}$$

Given equations (4.2), (4.3) and (4.4) the constraints necessary for the preservation of local isometry can be restated entirely in terms of the kernel matrix **K** and input vectors x_i (equation (4.5)).

$$\|y_i - y_j\|^2 = \langle y_i, y_i \rangle - 2\langle y_i, y_j \rangle + \langle y_j, y_j \rangle . \tag{4.4}$$

$$k_{ii} - 2k_{ij} + k_{jj} = \|x_i - x_j\|^2 . \tag{4.5}$$

It is this formulation that is used in the final optimization problem of the semidefinite embedding algorithm given in the subsequent section.

4.1.5 Additional Constraints

Thus far, the problem of learning an isometric embedding for the input vectors x_i, where i $\in [1...n]$, has been constrained to preserve local isometry (equations (4.3) and (4.5)). The formulation chosen in equation (4.5) is expressed in terms of a kernel matrix **K**, where **K** is defined as the dot product between output vectors y_i, i $\in [1...n]$. Next, the problem is further constrained by centering all output vectors y_i about the origin as given in equation (4.6). This step is essential in order to reduce the final SDP problem by one degree of freedom (see section A.1). It is also necessary in order to simplify the optimization of the SDP (equation (4.10)) as will be discussed shortly.

$$\sum_{i=1}^{n} y_i = 0. \tag{4.6}$$

We see that equation (4.6) can also be expressed in terms of **K**, as **K** is defined to be a matrix of dot products between all output vectors y_i (equation (4.3)). We see that equation (4.6) can be restated as equation (4.7) by restricting the sum of all elements in the kernel matrix to equal zero.

$$\sum_{i=1}^{n}\sum_{j=1}^{n} k_{ij} = 0. \tag{4.7}$$

Thus far, the original problem of learning the underlying low dimensional and isometric manifold from a set of high dimensional data points, is restructured into a learning problem over the kernel matrix **K**. If **K** is known, extracting the lower dimensional output vectors in $\mathbf{Y}$, is a straightforward eigendecomposition of $\mathbf{K}$. The problem being defined in this manner allows greater restraint on the final solution for output vectors in $\mathbf{Y}$. This is due to the fact that semidefinite programming, as discussed in section A.2.2, allows the definition of constraints on the final solution of $\mathbf{K}$. Equations (4.5) and (4.7) characterize the matrix $\mathbf{K}$ in terms of desired properties in the low dimensional embedding. In addition to this, Weinberger and Saul explain that the matrix $\mathbf{K}$ must also maintain the following characteristics [87]:

- $\mathbf{K}$ *must be confined to only symmetric matrices.*
- $\mathbf{K}$ *must have nonnegative eigenvalues.*

As defined in A.1, a symmetric, real valued matrix having nonnegative eigenvalues is characteristic of a positive semidefinite matrix. This implies that our problem is restricted to an optimization over semidefinite matrices, and is formulated thereby as a semidefinite programming problem [8].
Bear in mind, that in order to construct this problem as an instance of a semidefinite program, there must exist an optimization over an unknown. Weinberger and Saul [87] chose to maximize the variance between output vectors in the low dimensional manifold. This ensures that the projected outputs are as far apart as possible, and thus *unfolding* the high dimensional manifold to a flat, and more spread out respresentation in lower dimensions. Maximizing the *projected variance* in the output vectors can be restated as maximizing the *square distance* between output vectors y_i. This is defined in equation (4.8).

$$Max(var(\mathbf{Y})) = Max(\frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{n}\|y_i - y_j\|^2). \tag{4.8}$$

By employing the centering constraint from equation (4.6), the constraint in equation (4.8) can be restated in terms of only the output vectors y_i, as given in equation (4.9).

$$Max(var(\mathbf{Y})) = Max(\sum_{i=1}^{n}\|y_i\|^2). \tag{4.9}$$

Again, we want to formulate this into a constraint on the kernel matrix $\mathbf{K}$. This is accomplished by using the definition of the kernel matrix (equation (4.3)), and the centering constraint in equation (4.6). We simplify equation (4.9) to maximizing the sum of all diagonal elements of the kernel matrix as given in equation (4.10). This is further simplified as maximizing the *trace* of the kernel matrix $\mathbf{K}$ (equation (4.11)). Therefore, the optimization of maximizing the variance or square distance between output vectors y_i, simplifies to maximizing the *trace* of the kernel matrix $\mathbf{K}$ as demonstrated in equation (4.10).

$$\begin{aligned} Max(var(\mathbf{Y})) &= Max(\sum_{i=1}^{n} \|y_i\|^2), \\ &= Max(\sum_{i=1}^{n} k_{ii}), \qquad (4.10) \\ &= Max(trace(\mathbf{K})). \qquad (4.11) \end{aligned}$$

4.1.6 Semidefinite Programming Problem

In summary, the semidefinite embedding algorithm as proposed by Weinberger and Saul [87] is formulated into the following semidefinite programming problem:

$$\textit{Maximize trace}(\mathbf{K}) \textit{ subject to:} \qquad (4.12)$$

$$\mathbf{K} \textit{ is a positive semidefinite matrix: } \mathbf{K} \succeq 0, \qquad (4.13)$$

$$\textstyle\sum_{i=1}^{n} \sum_{j=1}^{n} k_{ij} = 0, \qquad (4.14)$$

$$\textit{For all i,j where } a_{ij}^{x} = 1, k_{ii} - 2k_{ij} + k_{jj} = \|x_i - x_j\|^2. \qquad (4.15)$$

The ultimate goal of the semidefinite embedding algorithm, like other manifold learning algorithms, is to recover a low dimensional embedding of a high dimensional data set. The problem has now been reformulated as a semidefinite programming problem (equations (4.12) to (4.15)). We see that the trace of $\mathbf{K}$ is the selected maximization parameter, and given equation (4.3), the output vectors y_i are recovered from $\mathbf{K}$ in order to produce the desired low dimensional embedding. Recovering the outputs will be discussed in more detail in the next section. Maximizing the trace of $\mathbf{K}$, as discussed 4.1.5, essentially is aspiring to have output vectors that are as far apart as possible in the low dimensional embedding. In addition to this restriction, the output vectors are constrained to centering about the origin in equation (4.14), and to preserve local isometry in equation (4.15). The matrix $\mathbf{K}$ is also restricted to the class of semidefinite matrices in equation (4.13), thus requiring it to be symmetric, real and having only positive eigenvalues. It is important to note that Weinberger and Saul show in [87] that maximizing the trace of $\mathbf{K}$ is bounded. Therefore, the output vectors cannot be pulled infinitely apart. This is primarily due to the constraint in equation (4.15) preserving local distances, and the underlying assumption that the graph of all neighbourhoods is indeed a connected graph.

4.1.7 Recovering Outputs

Of course, the final step in this algorithm is to recover the output vectors y_i for i $\in [1...n]$. The embedding space $\mathbf{Y}$ is recovered from the eigenvectors and eigenvalues of the matrix $\mathbf{K}$. Let the n eigenvectors of $\mathbf{K}$, denoted by v_α where $\alpha \in [1...n]$, be established as the rows of $\mathbf{V}$. Let $v_{\alpha i}$ denote the i^{th} element of the α^{th} eigenvector, and let λ_α denote its respective eigenvalue. Therefore, each entry of the kernel matrix can be represented as follows:

$$k_{ij} = \sum_{\alpha=1}^{n} \lambda_\alpha v_{\alpha i} v_{\alpha j}. \tag{4.16}$$

Thus, a d dimensional embedding, which is locally isometric to the input vectors x_i, for i $\in [1...n]$, can be obtained as follows:

$$y_{i\alpha} = \sqrt{\lambda_\alpha} v_{\alpha i}. \tag{4.17}$$

Here, each y_i corresponds to an input vector x_i, where i $\in [1...n]$. The number of eigenvectors and eigenvalues of $\mathbf{K}$ that are necessary for a d dimensional embedding is equal to d. In this case $\alpha \in [1...d]$. For example if a two dimensional embedding is desired, each output vector y_i will have a component in

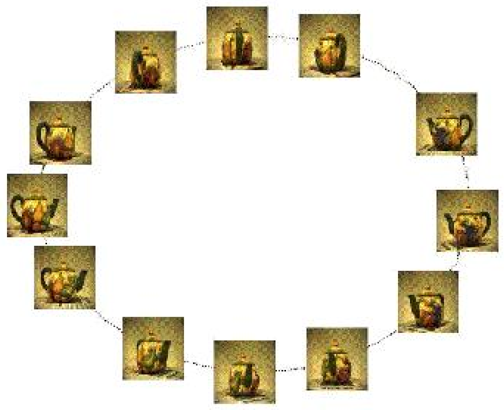


Figure 4.2: SDE maps the teapot images taken at 360^o of rotation to a circle where similarly angled images are displayed closer to one another in the circle. This image is taken from [87].

the first and second dimension in order to plot this point in two dimensions. This will result in y_{i1} and y_{i2} for all n output vectors. Although a two or three dimensional embedding may be desired, the *inherent* dimensionality of the data set is represented by the number of nontrivial eigenvalues of $\mathbf{K}$. Let the number of nontrivial eigenvalues of $\mathbf{K}$ be denoted as T. If the desired number of output dimensions is d, and $d << T$, there is loss of information in a d dimensional embedding of the data set. By examining the number of nontrivial eigenvalues of $\mathbf{K}$, one is able to assess the quality of a d dimensional embedding. We will use this sort of analysis in the next chapter, when comparing dimensionality reduction algorithms on microarray data sets.

4.2 Applications of Semidefinite Embedding

The authors of [87] applied SDE to many different image data sets where the number of input vectors n did not exceed 2000 images. Typically, the number of dimensions D, of the tested data sets was 250. In assessing the quality of the SDE two dimensional embeddings, Weinberger and Saul [87] considered the location of output vectors with respect to one another, and the number of nontrivial eigenvalues. The findings were that SDE successfully recovered patterns of information from the input data. In assessing the two dimensional output plots of the data sets, it is clear that similar images are plotted in close proximity, and dissimilar images are plotted further apart. For example, Weinberger and Saul [87] tested a data set containing random images of a teapot taken at 360° of rotation. We see the sample output in figure 4.2, where SDE accurately depicts the ordering in a low dimensional view of the original images. The two dimensional embedding of the images is essentially a circle and all n output vectors representing each of the n images are plotted, however only a few of the actual input images are displayed. We see that the output display of the images shows the teapot in correct order of rotation. Related or similar images are plotted nearer to one another and unlike images are the furthest apart.

4.2.1 Eigenvalue Spectra

In [87] SDE was applied to six image data sets in total and the two dimensional embedding of each data set was analysed. Two dimensional embeddings are extracted and examined in terms of the eigenvalues corresponding to each eigenvector. In figure 4.3 we see the percentages of all nontrivial eigenvalues from the SDE outputs of each image data set. From left to right, the chart displays the value in percentages of variance contained in the eigenvalues for each of the six images. Figure 4.3(a) indicates that the teapot images taken at 180° of rotation were completely characterized on a straight line in one dimension. Therefore, the first eigenvalue is nonzero and accounts for all the variance in the output data. Images (b,c,d) of figure 4.3 are also entirely captured in the top two eigenvalues, therefore there were only two nontrivial eigenvalues. To view the two dimensional SDE plots of the images in figure 4.3 please see [87]. Notice that images of figure 4.3(e,f) have a greater number of nontrivial eigenvalues. The principal variance of the data is still contained in the first eigenvalue which accounts for over 60%

of the variance. The remaining eigenvalues are decreasing in percentages, as is expected. It may appear that SDE does not correctly estimate the inherent dimensionality of the data due to the higher number of nontrivial eigenvalues. However, as noted by Weinberger and Saul [87], SDE actually performed with greater accuracy than other existing manifold learning algorithms such as PCA and LLE.

4.2.2 Synthetic Data

The authors of [87] also compare output embeddings from synthetic data sets, such as input vectors drawn from a two dimensional image of the letter *P*. This is an example of nonconvex data. Figure 4.4 shows the results of SDE and other dimensionality reduction algorithms on the synthetic data. The ability of SDE to output the image *P* almost exactly, demonstrates that SDE is not foiled by nonconvex data. The HLLE (section A.5) algorithm of Donoho and Grimes [21] displays the image of *P* with similar accuracy. Weinberger and Saul compare the eigenvalue spectra of SDE to other linear and nonlinear dimensionality reduction methods and find that SDE consistently shows greater accuracy in determining the inherent dimensionality of the data set [87]. Also, due to the nature of the semidefinite programming problem defined for SDE (equations (4.12) to (4.15)), the constraint set essentially guarantees that the solution will lead to an embedding that is locally isometric to the original data set. Such guarantees are not well defined for other nonlinear dimensionality reduction methods [87].

4.2.3 Drawbacks

The main drawbacks of the SDE algorithm, as mentioned by Weinberger and Saul [87], is the fact that SDE is comparatively slow in computation time. The computational intensity required by SDE exists primarily due to the nature of the semidefinite programming problem in learning the kernel

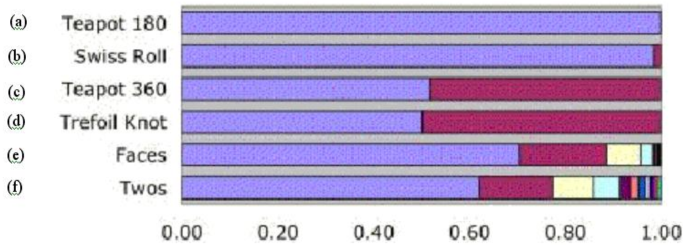


Figure 4.3: Spectrum of eigenvalues from SDE outputs on given image data sets. Figure (a) is completely characterized in a single dimension of SDE low dimensional outputs. Figures (e) and (f) indicate that greater than two dimensions are required to fully represent the variance in the high dimensional data sets. However, when comparing to other algorithms, SDE comes closest to correctly estimating the underlying dimensionality of the data sets. This image is taken from [87].

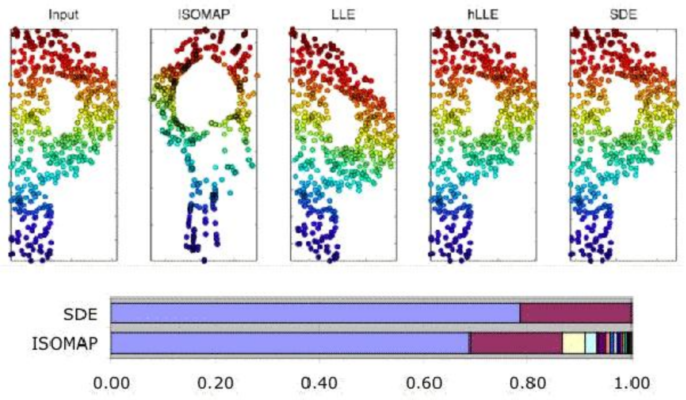


Figure 4.4: SDE almost exactly replots the input data, as does HLLE. Isomap and LLE result in some distortion to the original input data. SDE also exactly characterizes the inherent dimensionality of the output data. This is taken from [87].

matrix **K**. Also the SDE problem is not working with sparse matrices, therefore computation time is realistically too long for larger data sets. The work on semidefinite embedding was extended by Weinberger *et al.* in, *Nonlinear Dimensionality Reduction by Semidefinite Programming and Kernel Matrix Factorization* [86]. The authors attempt to factor the kernel matrix **K**, and allow its computation to occur in steps, rather than all at once. This decreases the computational overhead of calculations and allows the basic SDE algorithm to work with larger data sets. The improved algorithm is called *Landmark Semidefinite Embedding* and is described in the next section.

4.3 Landmark Semidefinite Embedding

As was discussed in 4.1.4, the primary goal of the semidefinite embedding algorithm is to learn the kernel matrix **K**, from which the output coordinates **Y** can be derived. However, running the SDE algorithm is computationally intense for large data sets, and generally only works well for data sets having 2000 inputs or less. Weinberger *et al.* expand their work on semidefinite embedding to incorporate larger data sets [86]. It is shown in [86] that for a well sampled manifold M, the derivation of the kernel matrix **K**, can be factored into a product of smaller matrices. A manifold is *well sampled* when distances between neighbours are small enough for the Euclidean approximation to be accurate [30]. As defined in the SDE algorithm (section 4.1), the kernel matrix **K** is an $n \times n$ matrix formed from the dot products of all pairs of output vectors y_i. The landmark semidefinite embedding algorithm (lSDE) defines a matrix of landmarks in a similar manner. The landmark set is a set of m selected input vectors taken from the original input data where $m << n$. The landmark set can be a randomly selected subset of input vectors, or a set that is chosen in some ordered manner. The matrix **L** is defined to be an $m \times m$ matrix of dot products between landmark output vectors. Weinberger *et al.* define **Q** to be an $n \times m$ linear transformation that approximates the entire data set from the landmark set. Therefore, the factorization of **K** in the *l*SDE algorithm is an approximation, and is given in equation (4.18):

$$K \approx QLQ^t. \tag{4.18}$$

The $n\times n$ kernel matrix **K** can be approximated by a much smaller $m\times m$ matrix **L** and its linear transformation **Q**. The integration of factorization in equation (4.18) into the SDE algorithm was seen to results in order-of-magnitude reductions in computation time. The factorization allows the possibility of studying much larger problems within the SDE framework. The idea of landmarks was first presented in [20], where landmarks were used to enhance the computational performance of MDS in the Isomap algorithm. Landmark Isomap is briefly discussed in section 3.4.2 of this thesis. The idea of landmarks is also the topic of discussion in [56] where landmarks are applied to accelerate the embedding of points in a sparse similarity graph. Intuitively, the *l*SDE algorithm is based on mapping high dimensional input vectors to a low dimensional space based on their distances to the selected set of landmarks. The main attraction of the landmark approach is the fact that the derivation of a large $n\times n$ matrix **K** can be broken up into solving a much smaller submatrix **L**, and subsequently using a linear transformation on the submatrix in order to map back to the embedding space. The number of landmarks m is a free parameter of the problem as is the number of k nearest neighbours. Kernel matrix factorization (equation (4.18)), is the fundamental difference in the *l*SDE algorithm as compared to the original SDE algorithm. A breakdown of how the kernel matrix **K** is factorized into equation (4.18) is given step by step in the subsequent section [86].

4.3.1 *l*SDE Algorithm

1. Given m randomly selected input vectors or *landmarks* from the data set, construct a linear transformation for approximately reconstructing the remaining *(n-m)* input vectors of the data set. Let m landmark vectors be denoted by u_r, where $r \in [1...m]$, and the reconstructed input

vectors by $\hat{x}_i$ for $i \in [1...n\text{-}m]$. Each of the reconstructed input vectors is defined as follows:

$$\hat{x}_i = \sum_{r=1}^{m} Q_{ri} u_r. \tag{4.19}$$

2. $\mathbf{Q}$ is an $n \times m$ linear transformation derived from a sparse weighted graph G defined in steps 3 and 4 [86, 56].

3. G is defined as follows:

   - Let the n nodes of G represent the input vectors in $\mathbf{X}$, including the m selected landmarks.
   - Weights are assigned to the input vectors and are used to propagate the landmark points towards the n-m points that are not in the landmark set. Let the matrix of weights be denoted by $\mathbf{W}$.

4. The matrix of weights $\mathbf{W}$, can be determined by reconstructing each input vector as a weighted sum of its k nearest neighbours. Therefore, $\mathbf{W}$ is found by minimizing the following error function:

$$E(\mathbf{W}) = \sum_{i=1}^{n} \left| x_i - \sum_{j=1}^{k} w_{ij} x_j \right|^2 . \tag{4.20}$$

   This is subject to the constraints that the row sums of $\mathbf{W}$ must equal 1 as this ensures that the weights are not affected by the choice of origin for the input vectors [86]. Also for every entry in $\mathbf{W}$ where inputs x_i and x_j are not neighbours, $w_{ij} = 0$.

5. Given the derivation of $\mathbf{Q}$ (steps 1-4), we proceed with the derivation of the embedding points in $\mathbf{Y}$. Let the set of landmark output vectors l_r, for $r \in [1...m]$, represent the set of low dimensional embeddings after the application of SDE to the initial m landmark input vectors.

6. The landmark output vectors l_r are used to derive a complete set of embeddings for the reconstructed input vectors $\hat{x}_i$, where $i \in [1...n\text{-}m]$.

Thus the set:

$$\hat{y}_i = \sum_{r=1}^{m} Q_{ri} l_r, \tag{4.21}$$

is the set of low dimensional embeddings for $\hat{x}_i$.

7. Next, we examine the output sets assuming that the m landmark output vectors are included in the set of all output vectors. Let the set of output vectors $\hat{y}_i$, for $i \in [1...n]$ be represented by $\hat{\mathbf{Y}}$. Recall from section 4.1.2 that the original output set from the SDE algorithm, y_i where $i \in [1...n]$, is represented by $\mathbf{Y}$. In the lSDE algorithm, Weinberger *et al.* [86] take $\hat{\mathbf{Y}}$ to be a sufficiently close to $\mathbf{Y}$. Therefore, it is understood that $\hat{\mathbf{Y}} \approx \mathbf{Y}$.

8. Let $\mathbf{L}$ represent an $m \times m$ matrix of dot products of all pairs of landmark output vectors l_r, $r \in [1...m]$ (equation (4.22)). Therefore, the matrix $\mathbf{L}$ is defined in a similar manner to the matrix $\mathbf{K}$ (equation (4.3) from section 4.1.4).

$$l_{rt} = \langle l_r, l_t \rangle\,, \ \textit{where } r,t \ \in [1...m]\,. \tag{4.22}$$

We see that kernel matrix factorization in the lSDE algorithm (equation (4.18)) follows if we make the following approximation for each entry in the kernel matrix $\mathbf{K}$:

$$k_{ij} = \langle y_i, y_j, \rangle \approx \langle \hat{y}_i, \hat{y}_j \rangle \ \textit{ where } i,j \ \in [1...n]\,. \tag{4.23}$$

The final mechanics of lSDE are very similar to the original SDE algorithm. The problem is formulated as a semidefinite programming problem similar to that in equations (4.12) to (4.15). The primary difference being only the factorization of $\mathbf{K}$ to get the approximation of QLQ^t. Therefore, given this factorization, the semidefinite program for the lSDE algorithm is as follows:

$$\textit{Maximize trace}(QLQ^t) \textit{ subject to:} \tag{4.24}$$

$$\mathbf{L} \textit{ is positive semidefinite: } \mathbf{L} \succeq 0 \tag{4.25}$$

$$\textstyle\sum_{i=1}^{n} \sum_{j=1}^{n} (QLQ^t)_{ij} = 0 \tag{4.26}$$

$$\text{For all } i,j \text{ where } n_{ij} = 1,$$
$$(QLQ^t)_{ii} - 2(QLQ^t)_{ij} + (QLQ^t)_{jj} \leq |x_i - x_j|^2 \tag{4.27}$$

We see that **K** is replaced by QLQ^t from equation (4.12) to (4.24). Also the final *equality* constraint in equation (4.15) is replaced by the *inequality* in equation (4.27). This is due to QLQ^t being only an approximation to the original kernel matrix **K**. Therefore, to ensure feasibility of the final solution, the distance constraints are relaxed to the inequality form. In practice, Weinberger *et al.* did not find this relaxation to be a great impedance to performance.

A further optimization in the *l*SDE algorithm is the number of constraints imposed on the semidefinite programming problem. As is noted by Weinberger *et al.*, there is redundancy in the constraint set passed to the original semidefinite program for SDE (equations (4.12)-(4.15)). The large number of constraints can be another bottleneck in SDE performance. In the *l*SDE implementation only a subset of the original constraints is applied to the semidefinite programming problem (equations (4.24)-(4.27)). Once a solution is found, it is tested against all original constraints. Any constraint being violated leads to another attempt at the solution for the semidefinite program. However, the next iteration will include the violated constraint(s). Again, it is noted in [86] that this iterative constraint process certainly improves computation time and does not compromise results. It should also be noted that the constraints on centering about the origin and restricting the matrix **L** to be positive semidefinite (equations (4.26) and (4.25)) are always applied to the semidefinite programming problem for *l*SDE.

In summary the *l*SDE algorithm directly implements the semidefinite programming problem from the original SDE algorithm with the kernel matrix factorized into QLQ^t. Given m landmark input vectors, the linear transformation matrix **Q** can be derived from a sparse weighted graph of the input vectors while minimizing the error function in equation (4.20). The matrix **L** is the solution to the semidefinite programming problem for lSDE (equations (4.24)-(4.27)). Subsequently the landmark output vectors l_r, $r \in [1...m]$, can be recovered from $\mathbf{L}$. Lastly the remaining set of output vectors $\hat{y}_i$, for $i \in [1...n\text{-}m]$, can be deduced using equation (4.21).

4.3.2 Findings: Landmark Semidefinite Embedding

In [87] the authors, Weinberger and Saul, described the theory and applications of the semidefinite embedding algorithm. It was noted that running the SDE algorithm on a data set which exceeds 2000 input vectors is computationally expensive, and requires impractical run times. A typical application would require on the order of several hours on a mid range desktop computer, running a pentium IV processor and average size RAM. The authors of SDE also discovered that a large portion of the constraint set passed to the semidefinite program are redundant or not explicitly required. The main objective behind *landmark* SDE was to reduce the computational burden of the original SDE alorithm, allow data sets to exceed the input size limitation, reduce the constraint set passed to the semidefinite programming problem, and not adversely affect the reliability of results. Weinberger *et al.* in their work on lSDE [86], test several large data sets with sizes ranging from 10000 to 60000 input vectors. It was found that the lSDE algorithm had significant impact on computation time. Given m, the number of landmarks ranging from 10 to 40, Weinberger *et al.* find lSDE took a fraction of the original SDE computation time. Weinberger *et al.* performed their testing in Matlab, using the SDP solver CSDP version 4.9 [11]. The authors also demonstrate that accuracy was sufficiently maintained from SDE to lSDE, and the computational time improvements were not heavily inhibitive to the reliability of results [86]. It is concluded by Weinberger *et al.* that the landmark approach to semidefinite embedding is a positive upgrade to the original algorithm, and successfully tackles the problems of computation time and input size.

4.4 Summary

In the preceeding sections we have seen the basic outline of the SDE and lSDE algorithms. As noted in section 4.2, there are certain advantages to the SDE approach over prior manifold learning or dimensionality reduction techniques. SDE is only concerned with local patches of the manifold. Therefore, in contrast to the Isomap algorithm, if the embedding space is non-convex the results are not flawed in any manner. SDE is able to determine the inherent dimensionality of the data set with greater accuracy than other methods. Also, the optimization in the final semidefinite programming problem (equations (4.12) to (4.15)) defines unfolding restrictions on the manifold in low dimensional space and this more accurately defines the output vectors y_i. The very nature of the semidefinite programming problem for SDE ensures the constraint set is satisfied in the final solution. This essentially guarantees that the low dimensional embedding will preserve local isometry due to the isometry constraints defined on every neighbourhood of the input space. The lSDE algorithm maintains all desirable properties of the original SDE algorithm, however it allows the kernel matrix to be determined in a more efficient manner and thereby heavily improves computation time. In this thesis, we incorporate the lSDE algorithm into a new method for dimensionality reduction of DNA microarray data. The KAS (*kernel alignment with semi-definte embedding*) algorithm, employs additional steps that are particularly suitable for microarray data. An outline of the KAS algorithm follows in the next chapter.

Chapter 5

KAS Algorithm and Experimental Methodology

The purpose of this study is to find an improved dimensionality reduction technique for DNA microarray data. We propose a novel method for dimensionality reduction of DNA microarray data which incorporates kernel alignment and feature selection in the semidefinite embedding algorithm. Our method, KAS *kernel alignment with semidefinite embedding*, is described in detail throughout this chapter. We begin with the motivation behind the *KAS* algorithm, followed by an overview of each step of the algorithm. This chapter concludes with a synopsis of the experimental methodology employed in this thesis.

5.1 Motivation

In chapter 2 we first introduced DNA microarray technology. When performing analysis of microarray data, often the size of the data sets and noise contained within the data sets is invasive and compromises precise qualitative and quantitative analysis. Once aquired, microarray data can be represented by an $n \times D$ matrix denoted by $\mathbf{X}$, where n is the number of genes (rows) across D samples. Each row in the data set represents the expression pattern of one gene, and each column represents a sample type or experimental condition. An input vector from the data set is considered to be an entire row, denoted by x_i; this represents the expression pattern for one gene across all experimental conditions. As discussed in section 2.3, dimensionality reduction methods are often applied to microarray data inorder to first reduce the size of the data set before further computational analysis and allow visualization in two dimensions. However, applications of dimensionality reduction techniques on microarray data have been only partially successful due to noise contained within the data and the large size of the data sets. Chao and Lihui [70] tested dimensionality reduction methods across the *samples* or columns of the microarray data set and subsequently used this for cancer classification. However, a standard objective of microarray data analysis is to better understand the *gene-to-gene* interactions that take place amongst the entire gene pool. Dimensionality reduction of microarray data has yet to effectively tackle the problem of finding a low dimensional embedding that provides an accurate visual representation of gene-to-gene interactions. Visual representations of gene relationships have been, at best, scattered and difficult to understand [61, 25]. As noted by Fellenberg *et al.* [25], in their work on microarray visualization techniques, often the problem in a two dimensional plot of microarray data is the unseparability of the data and the large number of genes. This renders a two dimensional diagram ineffective and difficult if not impossible to read. A problem with existing dimensionality reduction methods is the loss of information when projecting the data into two or three dimensions. Therefore, pre-processing steps that cater towards dimensionality reduction while maintaining accuracy, are greatly desired tools in the world of microarrays [61]. We attempt to deal with the current problems affecting dimensionality reduction methods by first pre-processing the data with feature selection and kernel alignment strategies. In our work, we endeavour to use semidefinite embedding (SDE) and landmark semidefinite embedding (lSDE), as tools for both the understanding of gene-to-gene interactions, and the replacing of a large unmanageable data set with one of much lower dimensionality. SDE and *l*SDE are discussed at length in chapter 4. First, we work with the microarray data set and extract its relevant features, as described in more detail in the next section. Next, we apply an adaptation of the kernel alignment algorithm, first proposed by Cristianini *et al.* [19], to a kernel matrix of input vectors. Subsequently, the aligned inputs will undergo dimensionality reduction by *l*SDE. Following the application of landmark semidefinite embedding on the aligned inputs, we have a final two dimensional plot of genes that maintains a high degree of accuracy and allows visual assessment of gene-to-gene relationships. As recommended by Weinberger and Saul [86], for the larger data sets that were tested in this thesis, we use the *l*SDE algorithm. This is done in order to ensure reasonable computation times that are competitive with other dimensionality reduction methods such as PCA and LLE. Our method, KAS provides an improvement over conventional algorithms on the data sets that were tested. We first briefly describe the techniques employed in our algorithm. The results of our experimentation is discussed in chapter 6.

5.1.1 Feature Selection

Feature selection of microarray data is a common approach in order to extract the relevant features or genes in the data set. Relevance is determined by genes being highly associated with one particular experimental class over other experimental classes. Feature selection is a tool that can be used to reduced the noise contained within microarray data. For the purposes of this study, the microarray samples will have two classes. In our study microarray data may consist of *normal* vs *tumorous* samples or *type1* vs *type2* cancer samples. The samples are classified as either being positive or negative. A feature selection strategy was tested by Golub *et al.* [31] in their work on leukemia classification. We employ this feature selection algorithm in order to determine which genes are top ranking. The leukemia data set tested by Golub *et al.* contained acute myeloid and acute lymphoblastic samples. These samples were labeled as either **AML** or **ALL**. The data sets we use are discussed further in chapter 6. The genes are indexed by i, where $i \in [1...n]$. Let μ_i^+ and μ_i^- refer to the mean of all positive(+) and negative(-) expression levels given for each gene. It follows that σ_i^+ and σ_i^- are the corresponding standard deviations. We employ the weight function given in equation (5.1). The weight assigned to each gene, or input vector x_i, is given as w_i:

$$w_i = \frac{\mu_i^+ - \mu_i^-}{\sigma_i^+ + \sigma_i^-}. \tag{5.1}$$

Input vectors having very *high* weight values show an association to the positive samples. Input vectors with very *low* weight values are strongly associated with the negative samples. The idea of feature selection is to trim away, from the original data set, genes that are not showing high association to either positive or negative samples. If we desire to have the top t ranking features of the data set, we select the highest *t/2* positive weights and lowest *t/2* negative weights. In addition, we employ the features selection algorithm of Golub *et al.* [31], to assign a labeling to the gene pool that remains. For example, genes with an association to positive samples will be assigned to the *+1* group. Genes having an association to negative samples are assigned to the *-1* group. Labeling is a necessary step for performing the next step in the KAS algorithm, *kernel alignment reduction*, which is discussed in the next section.

5.1.2 Kernel Alignment

In recent years we have seen the formulation of many new kernel based learning algorithms. Successful application of kernel-based algorithms is demonstrated in numerous fields including optical pattern recognition, text categorization and gene expression profile analysis to name a few [46]. This wide range work with kernels has lead to the recent concept of *kernel alignment*, first proposed by Cristianini *et al.* [19] in 2001. Given that a kernel is a representation of the original input data in higher dimensions [46], *kernel alignment* is a proposed measure of the degree of agreement between a kernel matrix and the given data set [19]. Let us begin with a matrix of input vectors, denoted by $\mathbf{X}$, as discussed in section 2. The expression pattern of each gene in $\mathbf{X}$ is represented by one row or input vector x_i. Let $\mathbf{u}$ represent the column vector assigning each gene in $\mathbf{X}$ to class *+1* or class *-1*. Therefore, for all $i \in [1...n]$, each element of $\mathbf{u}$, is denoted $u_i \in [+1, -1]$. The outer product Gram matrix of the label vector is defined as $\mathbf{M} = \mathbf{uu^t}$. Also, we define a kernel matrix $\mathbf{P}$ to be comprised of the dot products between each of the input vectors in $\mathbf{X}$. Therefore, each entry of $\mathbf{P}$ is expressed in equation (5.2) where $i,j \in [1...n]$.

$$p_{ij} = \langle x_i, x_j \rangle \,. \tag{5.2}$$

The computation of kernel alignment is taken as the correlation between the kernel matrix $\mathbf{P}$, and the matrix $\mathbf{M}$. Let the *Frobenius Product* [19] between two matrices be defined as $\langle \mathbf{P}, \mathbf{M} \rangle_F = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} m_{ij}$. As defined by Cristianini *et al.* [19], the measure of alignment A between the two matrices is given in equation (5.3).

$$A(\mathbf{P}, \mathbf{M}) = \frac{\langle \mathbf{P}, \mathbf{M} \rangle_{\mathbf{F}}}{\sqrt{\langle \mathbf{P}, \mathbf{P} \rangle_{\mathbf{F}} \langle \mathbf{M}, \mathbf{M} \rangle_{\mathbf{F}}}} \tag{5.3}$$

Our method, *KAS* employs a reduction algorithm that we will call *kernel alignment reduction*. The methodology was first proposed by Thomas [73]. Let $P(\mathbf{x_i}, *)^{\mathbf{t}}$ represent the i^{th} row in the kernel matrix $\mathbf{P}$. This is shown in equation (5.4). Using $(u_i\mathbf{u})^{\mathbf{t}}$ to represent the label vector times the i^{th} label, the alignment value a_i, of each individual input vector $\mathbf{x_i}$, is given by equation (5.5). Each a_i is thus dependent upon the i^{th} row of both $\mathbf{P}$ and $\mathbf{M}$.

$$\mathbf{P}(\mathbf{x_i}, *)^{\mathbf{t}} = (p_{i1}, p_{i2}, ..., p_{in})^t. \tag{5.4}$$

$$a_i = \frac{\mathbf{P}(\mathbf{x_i}, *)^{\mathbf{t}}(u_i\mathbf{u})}{\sqrt{\mathbf{P}(\mathbf{x_i}, *)^{\mathbf{t}}\mathbf{P}(\mathbf{x_i}, *)(u_i\mathbf{u})^{\mathbf{t}}(u_i\mathbf{u})}} \tag{5.5}$$

Kernel alignment reduction, as employed in *KAS*, aims to reduce the size of the data set based on the alignment values a_i. As discussed in section 2, we strive towards improving the effectiveness of dimensionality reduction on microarray data. One problem in visualization of microarray data is the inseparability of the genes in two dimensions. Groups or clusters are intermixed and the line of separation is difficult to assess. As noted by Thomas [73], when employing equation (5.5), we see that low alignment values represent genes or inputs that are bordering between groups or classes. In the context of microarray data, this is indicative of the fact that these genes are not strongly associated to either group. Therefore, we remove all such genes with low alignment values, and attempt to retain only the genes having strong association to others in the data set. The kernel alignment reduction algorithm involves three basic steps.

- Compute alignment values for all input vectors $\mathbf{x_i}$ where $i \in [1...n]$ (equation (5.5)).
- Remove the input vector with the lowest ranking alignment value, and reconstruct the matrices $\mathbf{P}$ and $\mathbf{M}$.
- Compute alignment values again for the remaining inputs.

Repeat steps 2 and 3, until all alignment values satisfy a lower bound. The final product, following kernel alignment reduction, is the kernel matrix $\mathbf{P}$ containing only highly aligned inputs. This matrix is then used by the lSDE algorithm in order to reduce the size of the original microarray data set, and provide a useful two dimensional plot of the data.

5.1.3 Summary of the KAS Algorithm

The semidefinite embedding algorithm and landmark semidefinite embedding was discussed at length in chapter 4. Building upon this, and the overview of feature selection and kernel alignment reduction given in sections 5.1.1 and 5.1.2, we are now ready to outline the full KAS algorithm.

- Given the input data matrix $\mathbf{X}$, first perform feature selection (section 5.1.1) of the top ranking features in the data set. Let the new data matrix be denoted by $\mathbf{F_x}$.

- Assign a label vector $\mathbf{u}$ to the features in $\mathbf{F_x}$.
- Perform kernel alignment reduction, as described in section 5.1.2, on the input matrix $\mathbf{F_x}$. Let the resulting matrix following kernel alignment reduction be denoted by $\hat{\mathbf{X}}$.
- Derive the lSDE embedding of $\hat{\mathbf{X}}$ and plot the two dimensional display of the outputs in $\mathbf{Y}$.

The computational demands of this algorithm primarily reside in the third step, where kernel alignment reduction is performed. However, in comparison to other methods, KAS has reasonable computation times. As we will see in upcoming chapters, the computational overhead is made up by the fact that KAS is able to perform dimensionality reduction with a higher degree of accuracy than the three other algorithms tested in this thesis. In the experimentation for this thesis, the KAS algorithm is tested on four different microarray data sets and rigorously compared with PCA, LLE and Isomap. The data sets are described in the next section. An overview of these algorithms can be found in chapter 3. For the remainder of this chapter we describe the experimentation process applied in this thesis.

5.2 Experimentation

The purpose of the experimentation described in this thesis is to test the dimensionality reduction capability of the *KAS* algorithm when compared to other conventional dimensionality reduction methods. The data sets used in this study are discussed in section 5.2.1. The experimental methodology follows in sections 5.2.2 and 5.2.3. An overview of the results is given in the next chapter.

5.2.1 Data Sets

The four microarray data sets that are used in the experimentation for this thesis are listed below. As discussed in section 3.2.1, each of the rows in the data sets represents the expression pattern of one gene and each column represents an experimental sample. The microarray data sets used in this thesis contain two types of samples. Typically one group of samples is *normal* and one group is *cancerous*. For experimentation purposes only two standard pre-processing procedures were applied to each data set. Thus, the data sets were normalised and scaled to have expression values all greater than zero. Such procedures were discussed in section 2.4. The microarray data sets used in this thesis are as follows:

1. Colon Cancer Data: The colon cancer data set was first tested by Alon *et al.* [1] in their work on colon cancer classification. The sampling was taken from colon adenocarcinoma tissues that were collected from patients. The test set employed in our testing consists of 22 samples where 8 are normal samples and 14 are cancerous samples. The number of genes in the data set is 2000.

2. AML/ALL Data: The acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL) data set was first used by Golub *et al.* in [31]. We used the combined leukemia data set consisting of 38 bone marrow samples obtained from adult acute leukemia patients at the time of diagnosis before chemotherapy. This data set measured expression levels across 7129 genes. The first 28 columns represent the ALL samples, the latter 10 columns represent the AML samples.

3. Prostate Cancer: The Prostate cancer data set was used by Singh *et al.* in [71]. In this data set, high-quality expression profiles were successfully derived from 52 prostate tumors and 50 nontumor prostate samples. These were obtained from patients undergoing surgery. This data set measures expression levels across 7216 genes and 102 samples

in total. Therefore, the first 50 columns represent normal samples and the last 52 columns represent tumourous samples.

4. Breast Cancer: The breast cancer data set was first studied by van't Veer *et al.* in their work on breast cancer classification [81]. For the purposes of our experimentation, we used a smaller breast cancer test data set that measured expression levels across 2000 genes and 19 samples. From these samples, 12 patients developed breast cancer and 7 remained disease free within 5 years of the study. Therefore, again the comparison is across cancerous and normal samples. This data set initially contained missing values which were filled in as an estimation based on the average value of the gene's expression profile across similar samples. The approximation algorithm for missing values is outside the scope of this thesis.

5.2.2 Determination of Results

The final product of the KAS algorithm is a two dimensional plot of the outputs in matrix $\mathbf{Y}$, where each output vector y_i is a label representing a corresponding input vector x_i. For purposes of this study, it is imperative to examine how well the KAS algorithm depicts the microarray data in two dimensions. We tested the effectiveness of the *KAS* algorithm on four microarray data sets, and compared results to those generated by three conventional techniques in dimensionality reduction. The three methods of comparison were selected as *Principal Components Analysis*(PCA) [60], which qualifies as a linear dimensionality reduction approach, *Local Linear Embedding*(LLE) [66] and Isomap [77], which are two typical techniques for non-linear dimensionality reduction. Typically, when assessing the accuracy of dimensionality reduction algorithms on image data, or synthetic data such as the *swiss roll*, it is sufficient to view the two dimensional plot of the output vectors [27, 77, 66]. One can visually assess the quality of the plots by analyzing the images and checking if similar images are plotted in close proximity, and dissimilar images are plotted further apart. An example of this is the application of SDE to teapot images taken at different degrees of rotation, seen in chapter 4 figure 4.2. The LLE, Isomap, SDE and lSDE algorithms have all been tested on image data sets [87, 86, 77, 66, 27]. When working with microarray data, the problem of evaluating the accuracy of dimensionality reduction methods is not quite as simple. One can view the outputs in two or three dimensions, however, many of the gene-gene relationships are unknown in advance. Therefore, one can not accurately assess the validity of an algorithm based only on visualizing the outputs. In order to test the relative accuracy of the low dimensional outputs, we used a measure called the *inherent dimensionality of the data set.* This is used as a measure in estimating the accuracy of a two dimensional embedding as given by the different algorithms. The inherent dimensionality of the data is discussed in the next section. In addition to the inherent dimensionality of the data in two dimensions, we also clustered the output from each algorithm. The clusters were examined and compared on the basis of clustering validation indices across all tested agorithms. Our exact clustering methodology is discussed below. In this thesis, the use of visual results is limited to the assessment and comparison of two dimensional embeddings generated by the four algorithms. There are further zooming in and out tools that may be applied to microarray two dimensional embeddings. However, for the purposes of our work we examine only the outputs that are generated by the dimensionality reduction algorithms and compare the embeddings based on this.

Inherent Dimensionality of Data

The output matrix $\mathbf{Y}$ contains the embedding coordinates for a d dimensional embedding. Often, as in our study, it is a two dimensionl plot that is desired from the dimensionality reduction methods. However, all of the dimensionality reduction techniques that were tested in this thesis allow for outputs y_i in as many as d dimensions, where $d \in [1...n]$. The eigenvalues of each dimension of the embedding provide an estimate of the percentage of variance retained in that dimension. In an approach that is similar to the methods employed by Weinberger and Saul [87] in testing the SDE algorithm, we also examined the eigenvalue spectra generated by the dimensionality reduction algorithms tested in this thesis. This allowed us to assess the quality of the two dimensional display as given by each algorithm, and indicates when the outputs are an accurate depiction of the original data set. Therefore, ideally the first two dimensions contain all of the variance of the data set, thus proving that the algorithm is very accurate in depicting the high dimensional data in two dimensional space. However, often, in the case of dimensionality reduction methods, this is not the case, and we saw varying percentages of variance maintained in many more dimensions. For purposes of our study, we used the eigenvalue spectra of the output data as a measure of the variance retained in each of the lower d dimensions. This provided an estimate of the percentage of variance maintained in each dimension of the output data contained in $\mathbf{Y}$. We compared the eigenvalue spectra of each method and interpreted this as an estimate of the accuracy retained in the low dimensional embedding space. The maximum allowable dimensions was set to ten due to the computational burden experienced in the Isomap algorithm. (Note: The one algorithm that does not allow this sort of analysis is LLE (see section 3.4.1). Therefore, LLE was left out of the *inherent dimensionality* results.)

Clustering Validation

A popular approach for biologists in examining DNA microarray data has been applying clustering to the data. This allows assessment of gene-to-gene groupings and relationships that exist within the data. Genes with like expression profiles across the samples will cluster together and genes with dissimilar expression profiles will cluster apart. In this study we applied two clustering algorithms that have been used on microarray data, to the four microarray data sets in section 5.2.1. K-Means and Gustafson-Kessel clustering were applied to the two dimensional outputs from the dimensionality reduction methods and comparisons were made across the clusterings. A detailed description of the clustering algorithms is given in [3]. We compared a set of four indices that help evaluate the clustering quality from these two algorithms. It is important to note that these indices are not directly related to output data from the dimensionality reduction methods, however they provide a useful measure in comparing clusterings of the output data. We compared clusterings of the original unreduced data sets to those of the dimensionalty reduced data and were able to derive meaning from the clustering indices. An underlying assumption that was made is that the unreduced data sets will cluster with more accuracy and thus higher validation scores than the reduced data sets generated by the dimensionality reduction algorithms. We examined an index set of four indices, as each index alone is not a reliable numerical representation of the clustering [3]. However, the indices together can serve as a useful measure in comparing the outputs across multiple clusterings. Two of the indices are only generated by a *fuzzy clustering* algorithm, therefore the Gustafson-Kessel (GK) clustering method allowed assessment of such indices. The set of four clustering indices that were examined in this thesis are taken from [3] and are listed below.

Clustering Validation Indices

K-Means and GK clustering are applied to the output data sets of the dimensionality reduction methods that were tested in this thesis. Let t represent the total number of clusters and c_i represent each cluster where $i \in [1, t]$. Let n represent the number of rows in the data set. We define u_{ij} as the membership indicator of data point j in cluster i where $u_{ij} \in [0, 1]$. Let w_j represent an input vector from the data set $\mathbf{W}$ that is clustered, where $j \in [1, n]$. We have V $= \{v_1, v_2, ...v_t\}$ representing the set of cluster centers. Therefore, $\|w_j - v_i\|$ is the distance between element w_j and the cluster center v_i in the clustering of $\mathbf{W}$. Also, $n_{min_{ij}}$ is the minimum distance between an element w_j and any of the clusters centers v_i.

1. The Xie and Beni's Index (XB) aims to quantify the ratio of the total

variation within clusters and the separation of clusters [3].

$$XB(t) = \frac{\sum_{i=1}^{t}\sum_{j=1}^{n}(u_{ij})^m(\|w_j - v_i\|)^2}{n_{min_{ij}}\,\|w_j - v_i\|^2} \tag{5.6}$$

The optimal number of clusters should minimize the value of the XB index. In this experimentation the value of m is set to 2. Although there is no logical arguement for this assignment to m, this has been chosen because a value of $m = 2$ has been commonly used in the literature [3, 32].

2. The Dunn index (DI) is ideally used to check for well separated and compact clusters [32]. Let x and y be input vectors from the data set **W** that is clustered. Let c_i, c_k *and* c_p represent the clusters where *i,k,p* $\in [1, t]$. We let $D(c_i,c_k)$ represent a dissimilarity function between the two clusters i and k. This dissimilarity function is defined in 5.7 as the minimum distance between x and y from two different clusters. Here $d(x,y)$ represents the distance between x and y.

$$D(c_i, c_k) = min_{(x\in c_i)(y\in c_k)} d(x, y) \tag{5.7}$$

Let $diam(c_p)$ represent the diameter of a cluster c_p. This is expressed as the maximum distance between any inputs x and y in the same cluster as given in equation (5.8).

$$diam(c_p) = max_{(x,y\in c_p)} d(x, y) \tag{5.8}$$

Therefore, the Dunn index for a clustering of t clusters can be expressed as follows:

$$DI(t) = min_i \left\{ min_{k,i\neq k} \frac{d(c_i, c_k)}{max_{p\in[1,t]} diam(c_p)} \right\} \tag{5.9}$$

The main drawback with direct implementation of the Dunn index is computational since calculating becomes computationally very expensive as t and n increase. Also the Dunn index is sensitive to the presence of noise within the data [32].

3. Bezdek [6] designed the partition coefficient (PC) to measure the amount of *overlap* between clusters. He defined the partition coefficient (PC) as follows.
$$PC(t) = \frac{1}{n} \sum_{i=1}^{t} \sum_{j=1}^{n} u_{ij}^2 \tag{5.10}$$
The optimal number of clusters will have a maximum value for this index.

4. Classification Entropy (CE) measures the fuzzyness of the cluster partition alone and is similar to the partition coefficient. When measuring the fuzzyness of the clusters, either PC or CE alone may be sufficient. However, we used both measures in our testing, in order to strengthen the results.
$$CE(c) = -\frac{1}{n} \sum_{i=1}^{t} \sum_{j=1}^{n} u_{ij} log(u_{ij}) \tag{5.11}$$

All experiments were performed in $Matlab^{TM}$ using the FuzzyClustering [3], LLE [66], Isomap [77] and LSDE [86] toolboxes.

KAS Parameters

The KAS algorithm as discussed in section 5.1.3 has parameters that need to be set before execution of the algorithm. These parameters are used for the *feature selection* and *kernel alignment* portion of the KAS algorithm. The first parameter, α, is needed for the feature selection step of KAS. This parameter defines in the algorithm the percentage of data that will be kept following feature selection. We generally set α to be between 80-90%, as the goal is to maintain the relevant features of the data while not sacrificing accuracy. We further preprocess the data in the kernel alignment reduction step of the KAS algorithm. The kernels are aligned and only input vectors with low alignment values are discarded. A low alignment threshhold value, β, is selected in order to ensure that a maximum of one third of the inputs may be discarded. In the experiments that were performed, on average for the AML/ALL and prostate cancer data sets, 15-20% of the inputs are left out due to low alignment values. The smaller data sets were aligned with a less restrictive value for β, where only 10% of inputs were discarded due to low alignment values. The number of nearest neighbours, k was set to three for all dimensionality reduction methods. We consistently maintained this value of k across all of the algorithms tested in this thesis, as the computational burden of experimentation rose significantly with increasing values of k. In their work with semidefinite embedding [87], Weinberger and Saul worked with a value of k set to four or five. There exists unreported experiments within our work where higher k values were tested, and we did not see a noticeable change in the results.

5.2.3 Biological Perspective

Before discussing the results, we would like to briefly examine the biologist's perspective and a few methods by which a biologist can currently examine DNA microarray data. A biologist is interested in using the dimensionality reduced data in order to better understand underlying relationships within the data. For example, a biologist may view a microarray data set in two dimensions and subsequently view the expression profiles across the genes in each cluster. An example of this is given in figure 5.2, for 16 clusters where all expression profiles for a given cluster are monitored in (a), and only the expression profile for the mean of each cluster is given in (b). Results from the experimentation in this thesis indicate that using the dimensionality reduced data set and subsequently clustering the data, can be used in a similar manner to the representation given in figure 5.2, thereby highlighting underlying relationships in the data. Also, we see in figure 5.1 (a) *Heirarchical Clustering* of a typical microarray data. This is a common method to again, divide the gene expression profiles into groups.

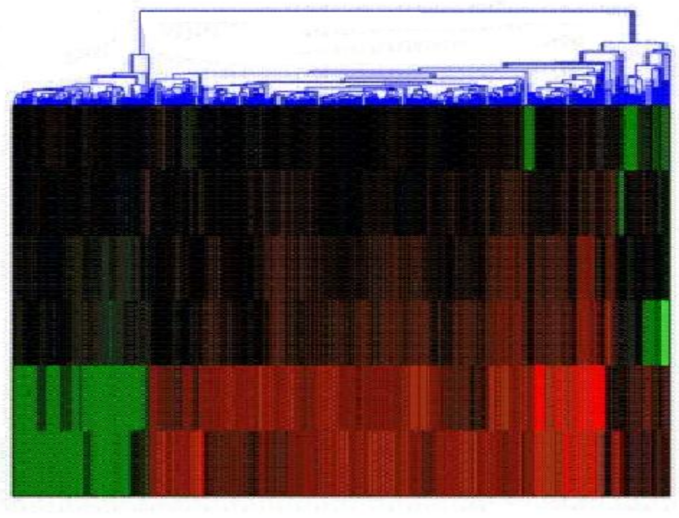


Figure 5.1: A view of gene expression data in a hierarchical clustering. Genes are clustered based on their expression profiles and grouped with like genes in the heirarchy at indicated at the top.

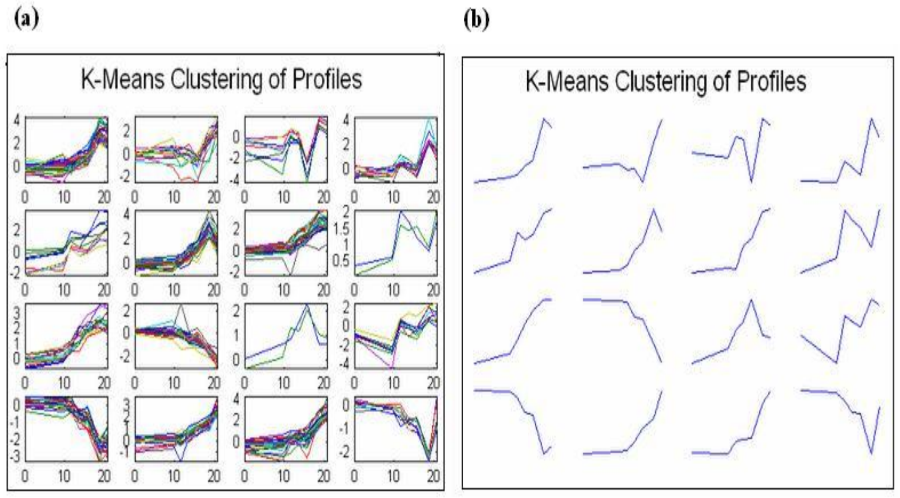


Figure 5.2: A view of gene expression profiles across 16 clusters. In (a), the profiles for each gene in the cluster is given, and only the mean profiles across each cluster is given in (b).

Chapter 6

Results

The list of experiments, data sets and the experimental methodology applied in this thesis were discussed in section 5.2 of the last chapter. The purpose of the experiments performed in this thesis was to test the dimensionality reduction capability of the *KAS* algorithm when compared to other conventional dimensionality reduction methods. The purpose of this chapter is to state the results of the experimentation without making any assumptions about the conclusions. The results of the tests conducted on each of the data sets is given in the sections that proceed. There is a summary of all the results provided in section 6.5. Chapter 7 will discuss the conclusions at greater length.

6.1 Results: Breast Cancer Data Set

The smaller data sets that were tested in this thesis are the colon and breast cancer data sets used by Alon *et al.* and van't Veer *et al.* respectively [1, 81]. The two dimensional outputs from each of the four algorithms is much clearer when working with the smaller data sets. We first tested the application of the four dimensionality reduction algorithms on the breast cancer data set and applied K-Means clustering to output data. This is given in figure 6.1. The outputs are plotted according to the first and second primary dimensions of the output data. Therefore, the x and y axes represent the first and second dimensions of the output vectors of each respective algorithm. In figure 6.1, the number of clusters, r was set to six. We did not find unexpected or significant changes with a varying number of clusters. We can see that the KAS outputs are well spread across two dimensions, and the division of the clusters is more easily detectable when comparing to the embedding of other algorithms. The KAS outputs in 6.1(a) allocate fewer genes to each cluster and do well in the separation of genes in two dimensions. The edges of each cluster are not closely connected to any group. We see that generally the output data from the four dimensionality reduction methods clusters well. LLE displays an embedding where the outputs are concentrated along one particular axis. This is a typical display of the LLE algorithm and can be seen in subsequent results. PCA and Isomap appear to have the most similarites in the two dimensional display of the data.

6.1.1 Clustering Validation Indices

The clustering validation indices from the K-Means and GK clustering algorithms are given in figure 6.2. In some of the clustering indices such as XB, there is inconsistent behaviour displayed by all algorithms, KAS, PCA, Isomap and LLE. This again is representative of the fact that not any one index alone is a reliable validation measure. However, we examine the indices for general behaviour patterns, particularly for KAS. As stated in section 5.2.2, we assume that the clustering of the unreduced data set will have more accuracy, and thus generate higher validation scores, than that of the dimensionality reduced data sets.

- Dunn Index: The DI scores in figure 6.2(a) show the breast cancer unreduced data set as having the highest or best DI score and LLE as having the lowest or worst DI score. The Dunn index is meant to evaluate the compactness and separation of the clusters. We saw the K-Means clustering in figure 6.1 applied to all dimensionally reduced

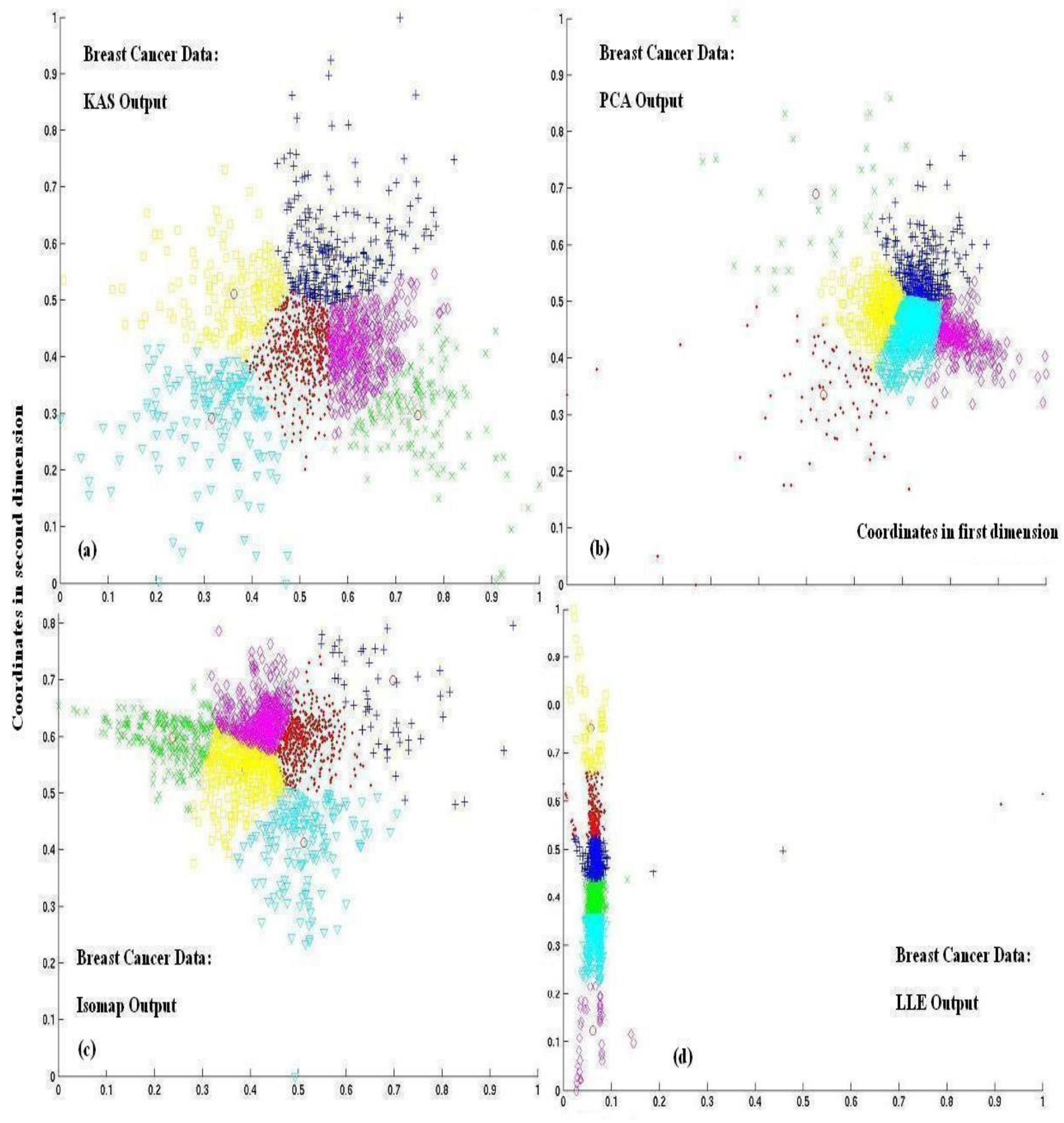


Figure 6.1: Two dimensional outputs of dimensionality reduction methods tested on Breast Cancer Data: The output of each algorithm was clustered with K-Means clustering. The circles within each cluster represents the cluster centre. KAS outputs show greater separation between bordering clusters. The number of clusters, r = 6.

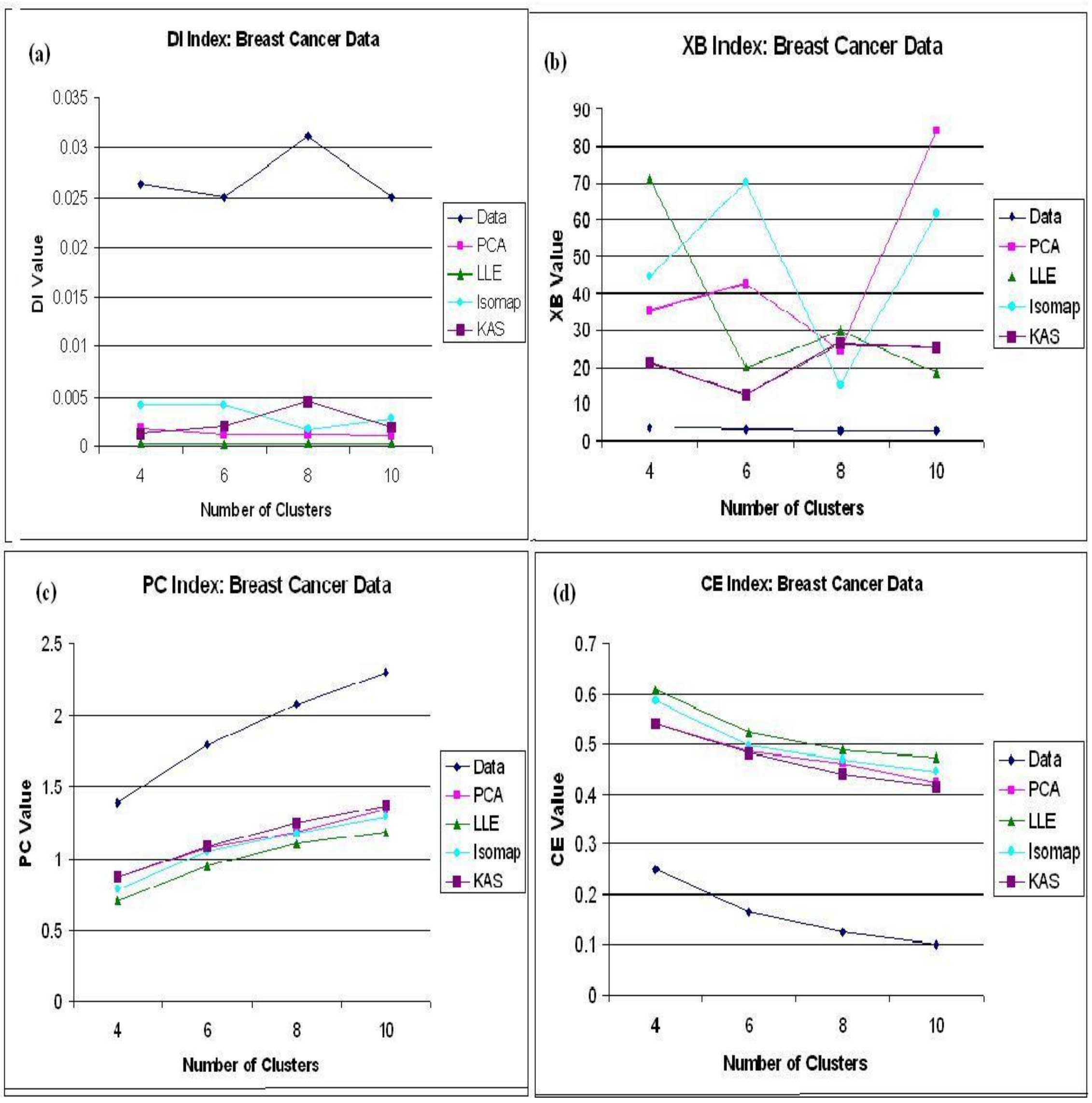


Figure 6.2: Breast Cancer Data: Clustering validation indices for K-Means and GK clustering of two dimensional output data. K-Means clustering is applied in (a) and (b) and GK clustering is applied in (c) and (d).

data sets. It is intuitive that the unreduced data sets would score well in clustering validation indices, as the number of data dimensions is much greater than only two. The clustering validation indices were evaluated on four to ten clusters. KAS, PCA and Isomap also have low DI scores. The KAS and Isomap DI scores are similar with an average difference of 0.0076. The KAS clusters follow the unreduced breast cancer data in general behaviour, as we see the value is higher for eight clusters and subsequently decreases for ten clusters. The unreduced data set peaks at eight clusters with a score of 0.031. KAS also peaks at eight clusters with a score of 0.046.

- XB Index: The XB index, again is a ratio of the total variation within clusters. The XB clustering index, will have the lowest value for an optimal number of clusters generated by the clustering algorithm. Although there is variation on XB scores across the clusterings in 6.2(b), we see that the unreduced breast cancer data set in 6.2(b) had the lowest XB score and KAS follows next. The unreduced data set has XB scores in the range of 2.84 to 3.74. KAS scores range from 12.66 to 24.41 and have the lowest range next to the unreduced data set.

- PC/CE Index: The partition coefficient and classification entropy indices try to assess the fuzziness or overlap between the clusters. Ideally, we would like crisp separation between the clusters, however often in reality this is not the case. The fuzziness of a clustering is highly related to the data set itself. As the number of clusters increases, the PC score will rise, and the amount of overlap between clusters decreases. We see in figure 6.2(c) that, next to the unreduced data set, KAS has the highest PC scores. Both KAS and the unreduced data set peak at ten clusters with PC scores of 1.37 and 2.3 respectively. The unreduced breast cancer data, again shows the best scores across all clusters. The PCA, Isomap and LLE algorithms all behave quite similarly. The PC scores are basically repeated in the CE index scores of 6.2(d). Here, however the lower CE score indicates a cleaner, less fuzzy partition. The unreduced breast cancer data has the best CE rating, followed closely by KAS, PCA, Isomap and LLE. It is intuitive that LLE would have a worse CE and PC score, as the two dimensional outputs are very compact and close together, thereby more conducive to overlap of the clusters. Again the CE scores are best at ten clusters when the unreduced data set has a score of 0.1 and KAS has a score of 0.41.

6.1.2 Variance in Two Dimensional Embedding

We see in figure 6.3 the eigenvalue spectra of the four algorithms on the breast cancer data set is given. The variance in the first dimension is given by the dark blue area at the very bottom of the bar graph. The subsequent eigenvalues for each dimension of output is given by the different shaded areas. The percentage of variance in the first dimension for the KAS outputs is 38%, which is highest amongst the algorithms. In this particular data set the PCA outputs have a relatively higher percentage of accuracy. The first dimension of the PCA outputs manages to retain approximately 37% of total variance of the data. We see that in two dimensions the KAS outputs do give a higher percentage of variance when compared to the three other algorithms. The LLE and Isomap outputs have lower percentages of variance retained in two dimensions for this particular data set as compared to the other tested data sets. Also, if we examine the total number of dimensions that retain a substantial percentage of overall variance, the KAS outputs show the least number of dimensions. This implies that the KAS outputs require a smaller number of total dimensions to preserve accuracy in a low dimensional embedding.

6.2 Results: Colon Cancer Data Set

Next, we performed a similar set of tests on the colon cancer data set. The two dimensional outputs of each algorithm can be seen in figure 6.4. K-

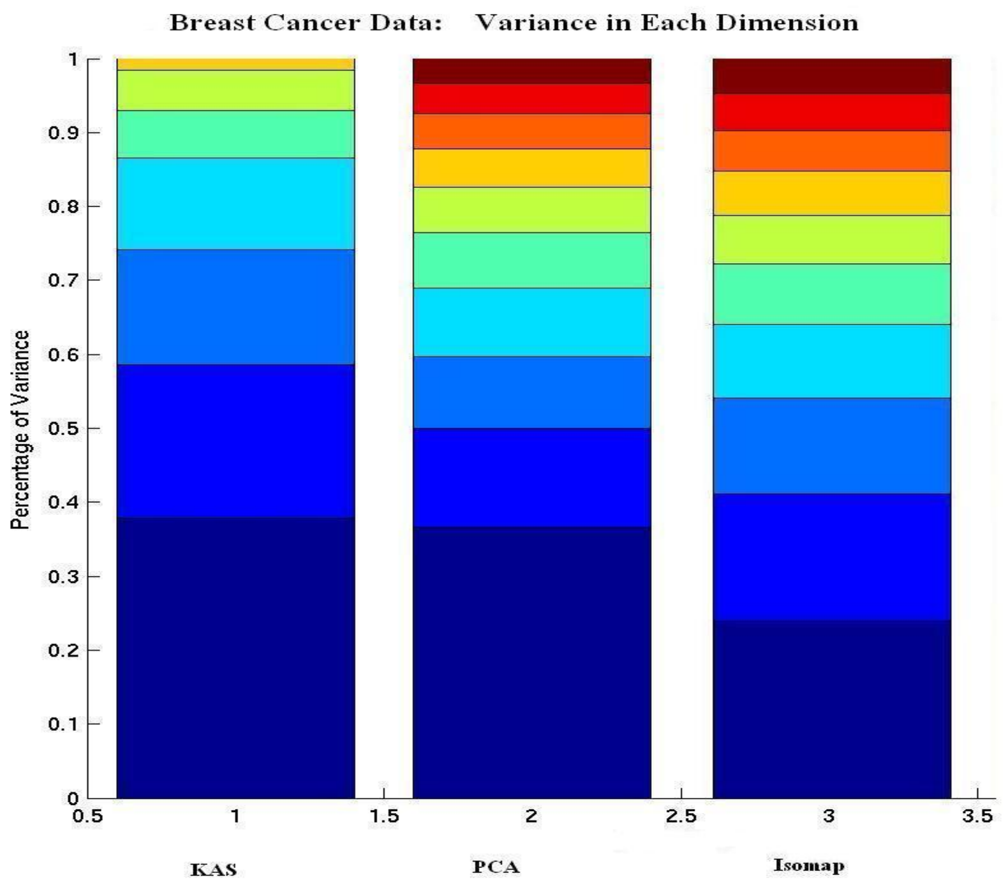


Figure 6.3: Comparison of variance in each dimension of output data generated by dimensionality reduction methods. First dimension is dark blue and begins at the bottom of the bar graph. KAS has the highest percentage of variance retained in two dimensions.

Means clustering is applied to each of the output data sets generated by the dimensionality reduction methods. In figure 6.4, the number of clusters, r was set to six. We see the PCA outputs are distributed across the two dimensions and more broad in size than other figures. The KAS outputs are also spread across the two dimensions, however to a lesser degree than the PCA outputs. The borders between clusters is easily discernible. The Isomap outputs are more bunched together in this figure than that of the breast cancer data set. LLE outputs in two dimensions and six clusters are not easily discernible and difficult to separate with only the human eye. We do see that the KAS outputs make for a clear visual display in two dimensions.

6.2.1 Clustering Validation Indices

The clustering validation indices, derived from the K-Means and GK clustering of the colon cancer data set, are given in figure 6.5. Again, we see that some of the clustering indices are useful for only general behaviour patterns as discussed below.

- Dunn Index: The DI scores in figure 6.5(a) shows the clustering of the Isomap output data and PCA output data as having the lowest or worst DI scores across the clusterings. For this particular data set the unreduced data set has scores that are very close to KAS. LLE begins with DI scores that are better than those of the unreduced data set. KAS and the unreduced colon cancer data set have DI scores that range from 0.0044 to 0.0065. For the number of clusters set to ten, KAS has the highest DI score. LLE outputs begin with high DI scores and fall back down to 0.0048 for ten clusters. Again, KAS outputs show similar behavior to the unreduced data set, which fairs well for the Dunn Index.
- XB Index: The XB clustering index, will have the lowest value for an optimal number of clusters generated by the clustering algorithm.

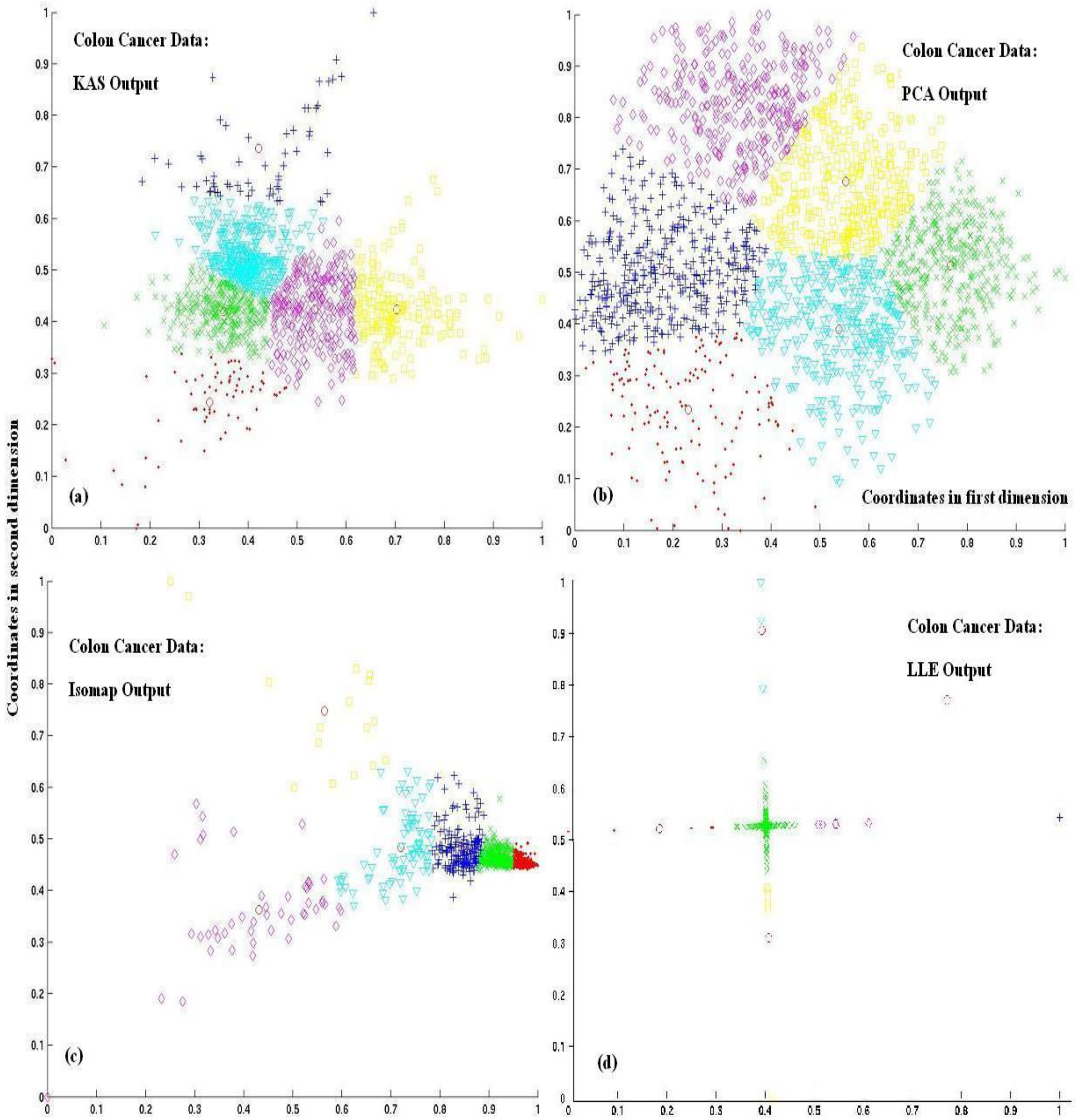


Figure 6.4: Two dimensional outputs for Colon Cancer Data. K-Means clustering is applied to the two dimensional data. The number of clusters, r = 6.

In this case, again the clustering of the unreduced colon cancer data provided the best XB index scores. The clustering of KAS output data follows that of the unreduced data set and indicates a general trend of good XB scores. KAS outputs have XB scores that range from 17.85 for $r = 4$ clusters, and end at 13.23 for $r = 10$ clusters. The Isomap outputs have a clustering that results in fair XB scores ranging from 18.76 to 19.82. PCA and LLE have XB scores that are not easily interpretable and jump drastically as the number of clusters changes. However, when the number of clusters is set to ten, both the PCA and LLE scores fall back down to 20.59 and 15.04 respectively.

- PC/CE Index: The partition coefficient and classification entropy of a clustering attempts to assess the fuzziness or overlap between the clusters. In figure 6.5(c) and (d) we see the two indices are close to the inverse of one another. In figure 6.5(c), the clustering of the unreduced colon cancer data attained the highest or best PC score. The unreduced data had scores ranging from 1.38 to 2.27 for the PC index. Similarly, in figure 6.5(b) the clustering of the unreduced colon cancer data gives the lowest or best CE score. Here the unreduced data set had CE scores in the range of 0.26 to 0.11. In both cases, the KAS outputs have the next best scores. The KAS scores range from 0.76 to 1.32 for the PC index and from 0.60 to 0.44 for the CE index. The PCA outputs, again have the next highest scores, following the KAS outputs. In this experiment, the Isomap and LLE clusters had the lowest PC and CE scores in comparison to the other algorithms.

6.2.2 Variance in Two Dimensional Embedding

Let us examine figure 6.6 where we see the eigenvalue spectra for three of the four tested algorithms on the colon cancer data set. KAS retained the highest percentage of variance in the first two dimensions which is close to 80%. Also, the top three dimensions in the KAS outputs account for

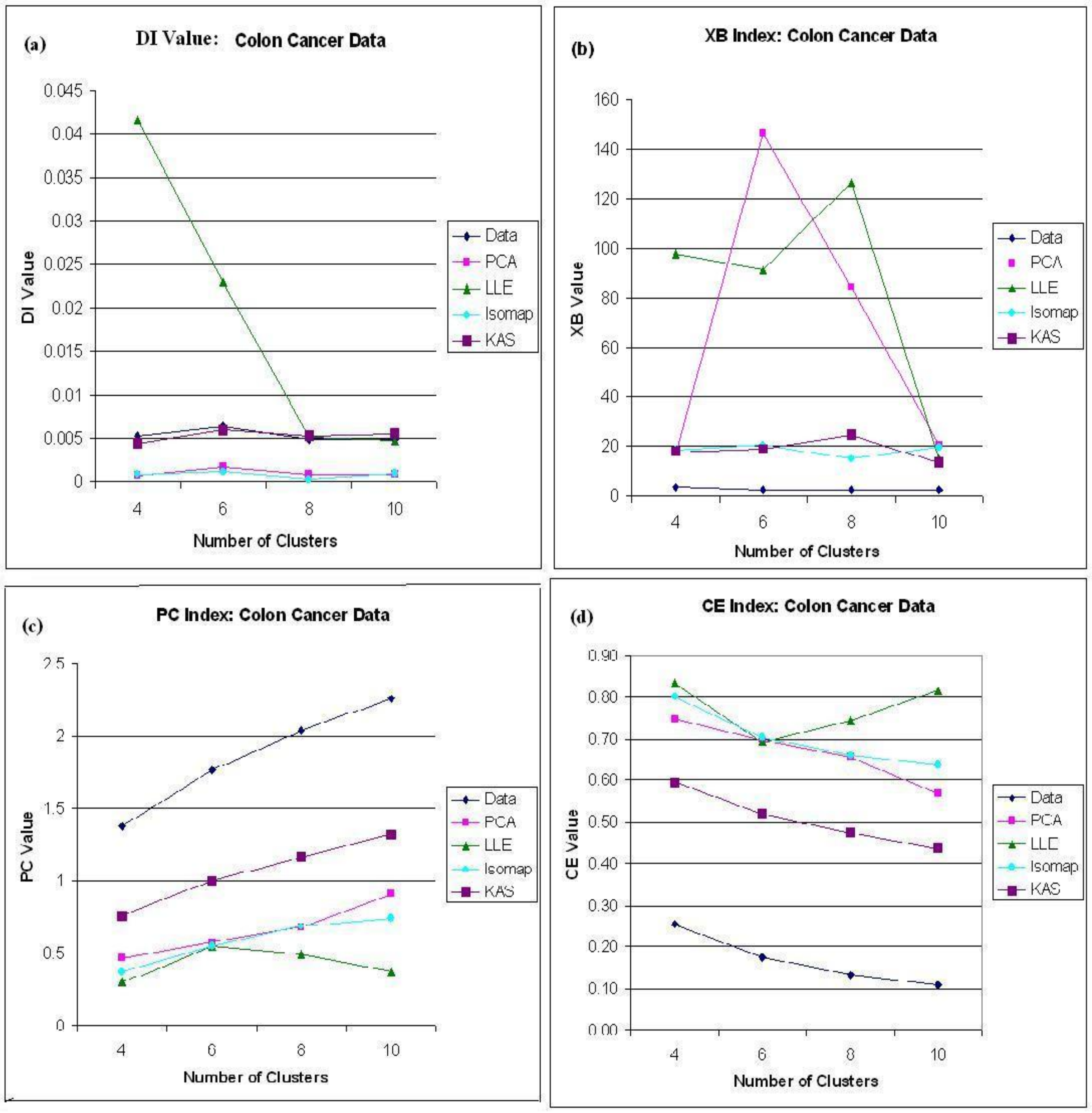


Figure 6.5: Colon Cancer Data: Clustering Validation Indices for K-Means and GK clustering of two dimensional output data of Colon cancer data set.K-Means clustering is applied in (a) and (b) and GK clustering is applied in (c) and (d).

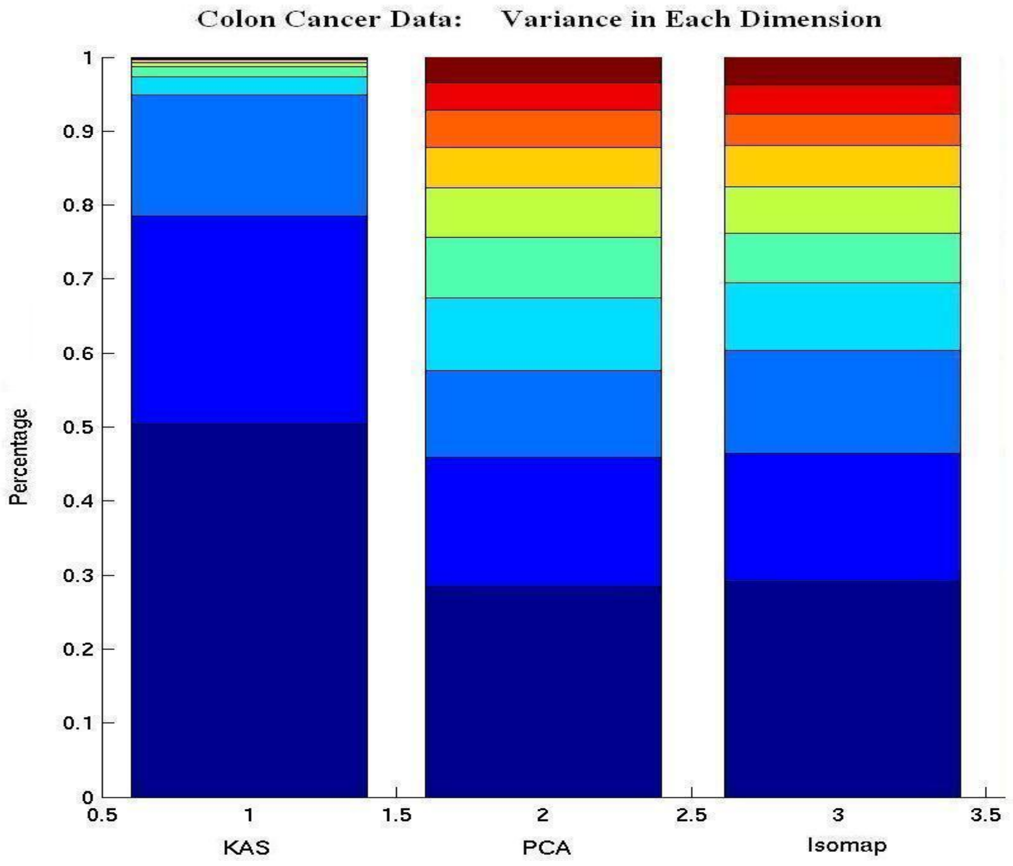


Figure 6.6: Comparison of Variance in each dimension of output generated by dimensionality reduction algorithms. First dimension is dark blue and begins at the bottom of the bar graph. KAS retains highest percentage of variance in two dimensions, which is close to 80% of total variance.

approximately 95% of the total variance in the data. This is significant especially in comparison to the other dimensionality reduction methods which require up to nine dimensions for the same percentage of variance. The PCA and Isomap outputs generally had similarities in eigenvalue spectra results for the colon cancer data set. PCA retains approximately 56% of variance in three dimensions and Isomap retains approximately 58% of variance in three dimensions. In only two dimensions of output, we see that the PCA and Isomap outputs are not able to retain even 50% of total variance contained in the high dimensional data.

6.3 Results: Prostate Cancer Data

In the experimentation for this thesis, two dozen tests were performed on the prostate cancer data set used by Singh *et al.* [71]. We tested across random subsets of the 7129 genes, each across 102 experimental conditions. Many tests with the entire data set were also performed. The two dimensional clustering results given in figure 6.8 were taken from experiments using the entire data set. In both the prostate data set and AML/ALL data set, the Isomap algorithm was replaced by the *landmark*-Isomap version of the original Isomap algorithm discussed in 3.4.2. In figure 6.8, K-Means clustering was applied to all output data generated by the four algorithms. We have selected a clustering where the number of clusters, r was set to eight. We see in figure 6.8(b), the PCA outputs display a typical PCA visual depiction. There is no significant concentration towards the centre, and the outputs form a circular shape in two dimensions. We can not easily assign a shape to the other diagrams. Again, they follow a similar pattern as seen in the other data sets. The KAS outputs, in figure 6.8(a) are more concentrated towards the centre, and the gene-to-gene distances increase as we move away from the centre. The LLE outputs are concentrated across three distinct directions and the Isomap outputs have less concentration of genes as we move away from the centre.

6.3.1 Clustering Validation Indices

The clustering validation indices are derived from the application of K-Means and GK clustering on the two dimensional outputs generated by each algorithm. We can see the results in figure 6.7. A clustering was also performed on the unreduced prostate data set, in order to compare. As expected, the index scores were highest for the unreduced data set.

- Dunn Index: The DI scores in figure 6.7(a), shows the clustering of the unreduced data set as having the highest and best DI scores across the clusterings. The LLE output data and KAS follow a similar pattern to that of the unreduced data set, whereby the DI scores drop for $r = 8$ clusters, and come back up for $r = 10$ and 12 clusters. KAS scores most closely resemble those of the unreduced data set. The KAS DI scores range from 0.0024 to 0.011 and the unreduced data set has DI scores ranging from 0.0031 to 0.0021. The Isomap output data gave the lowest DI scores and the PCA outputs rise in DI scores for the last two clusterings of $r = 10$ and $r = 12$.

- XB Index: The XB index had the best results for the unreduced data set, followed by the KAS outputs. A lower value for the XB index, is indicative of a good XB score. Also, the PCA outputs have good XB scores, close behind the KAS outputs. The KAS XB scores range from 32.16 to 30.37. The unreduced data set XB scores range from 4.61 to 3.41. A clear pattern of XB scores for the Isomap and LLE outputs is not obvious.

- PC/CE Index: To assess the PC and CE indices, we applied the GK fuzzy clustering algorithm to the two dimensional outputs generated by all four of the dimensionality reduction methods. In figure 6.7(c) and (d) we see the best scores are again held with the clustering of the unreduced prostate data set. The unreduced prostate data set had significantly better scores than the tested algorithms. KAS and PCA

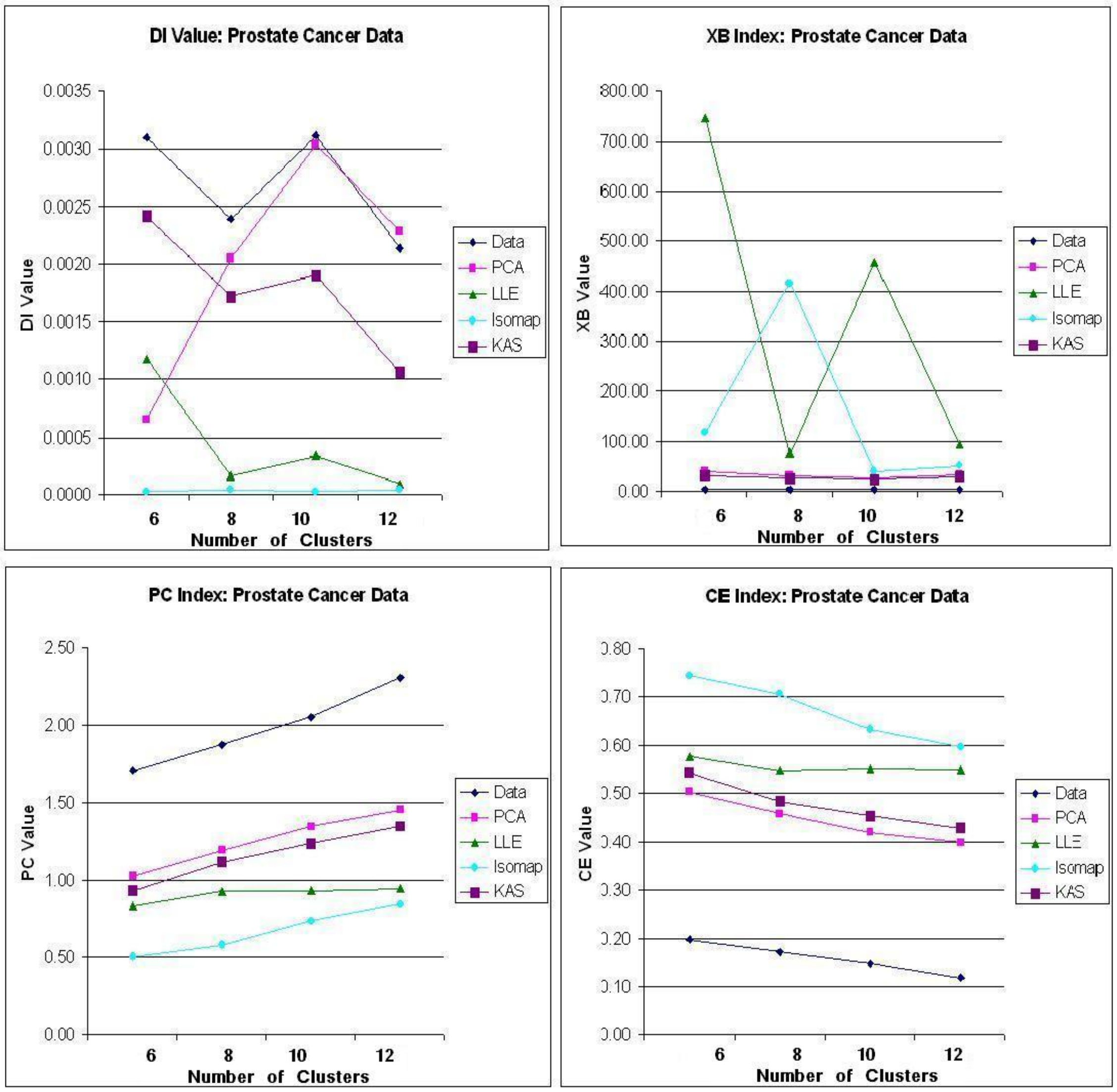


Figure 6.7: Prostate Cancer: Clustering validation indices for output data generated by all four dimensionality reduction methods.

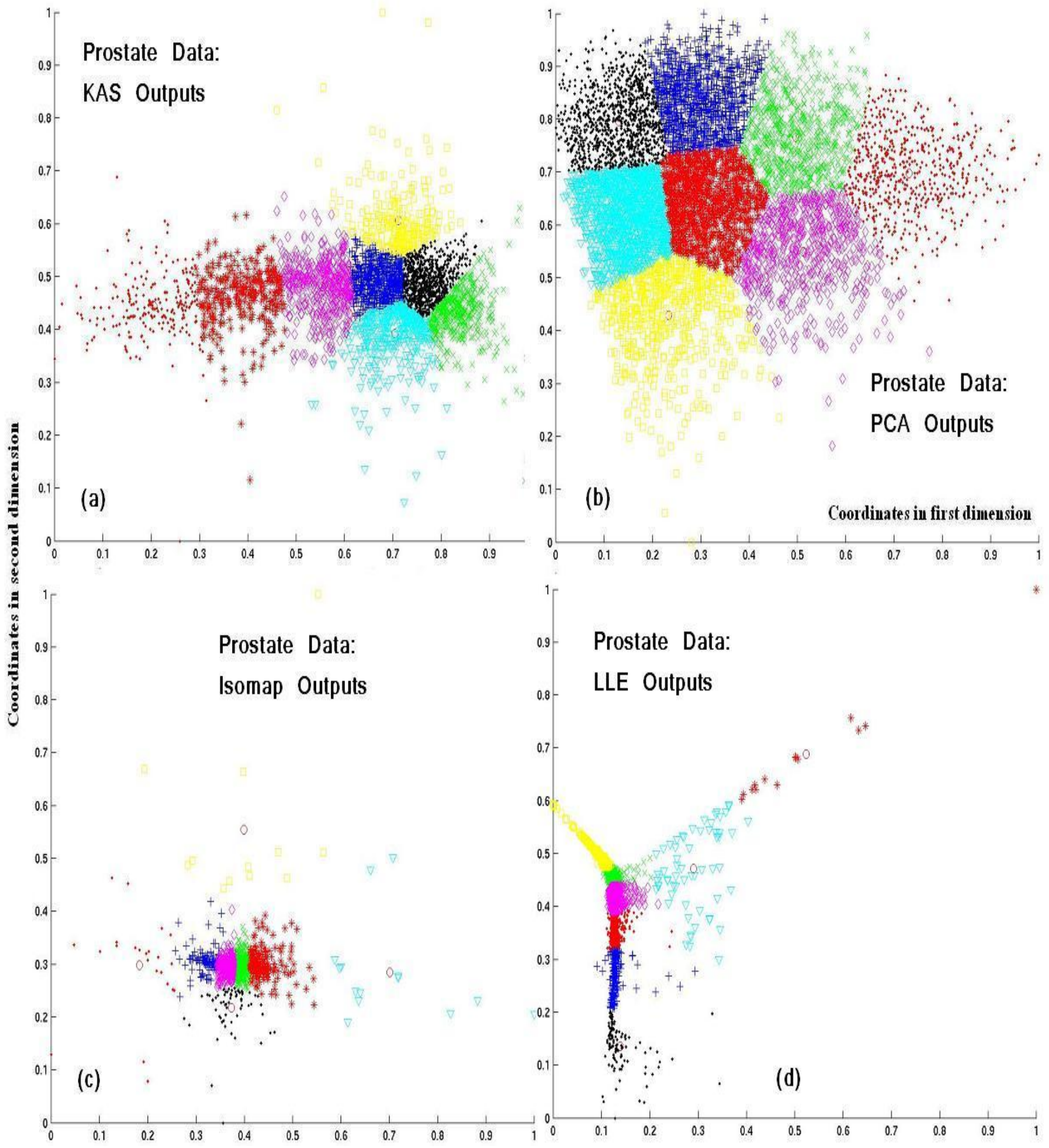


Figure 6.8: Two dimensional outputs for Prostate Data with K-Means clustering applied to the data. Here the number of clusters, r = 8.

follow the closest behind for both indices. In this case PCA scores were slightly higher than KAS scores for both indices. The two algorithms differ by an average score of 0.03 for the CE index and 0.27 for the PC index.

6.3.2 Variance in Two Dimensional Embedding

In figure 6.9 the eigenvalue spectra is given as a bar graph similar to the graphs in sections 6.1 and 6.2. The percentage of variance is given for each of the output dimensions. As we are interested in only two dimensions we hope the top two dimensions contain a large percentage of the total variance. For the prostate cancer data set, the KAS outputs retained approximately 62% variance in two dimensions. This rises to 72% in only three dimensions. The other algorithms need five or six dimensions of output data to attain results that are even close to this. In only two dimensions, we see that the PCA outputs retain approximately 40% of total variance and the Isomap outputs retain less than 30% of total variance.

6.4 Results: AML/ALL Data

In the experimentation for this thesis, approximately 20 tests were performed on the AML/ALL data set used by Golub *et al.* [31]. We tested across random subsets of the 7129 genes across 38 experimental conditions. Many tests with the entire data set were also performed. The two dimensional clustering results given in figure 6.10 were taken from experiments using the entire data set. In figure 6.10 we see the two dimensional embeddings from each of the dimensionality reduction algorithms tested in this thesis. The outputs were clustered using K-Means clustering with the number of clusters, $r = 10$ in this particular diagram. The visual results using a different number of clusters were very similar to figure 6.10. We see in figure 6.10(b) the PCA outputs are comparably the most spread out across two dimensions. It is difficult to derive meaning or assess gene-to-gene relationships from the data

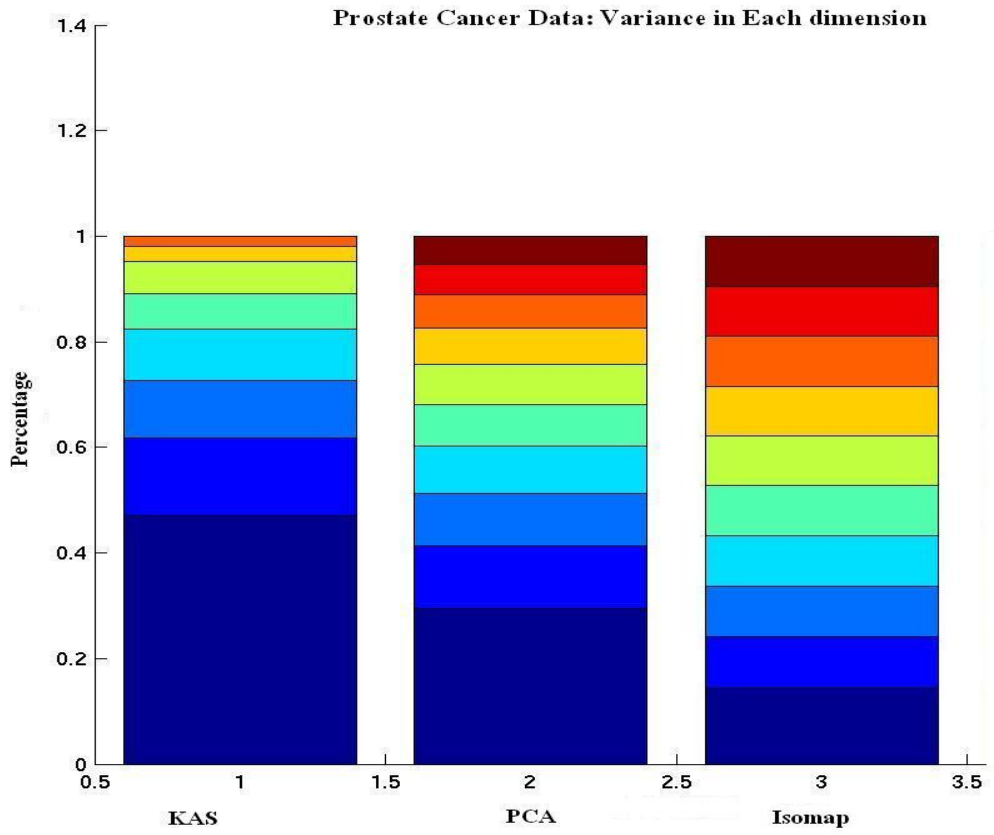


Figure 6.9: Comparison of Variance in each dimension of output generated by dimensionality reduction algorithms. First dimension is dark blue and begins at the bottom of the bar graph. KAS retains highest percentage of variance in two dimensions, which is close to 62% of the total variance contained in the unreduced prostate data set.

without clustering. KAS outputs are not so broadly spanned, however they show a clear view of the separation between clusters. The KAS outputs generally have fewer genes in each cluster, as the preprocessing steps trim away unnecessary genes. We saw similiar results with the prostate cancer data set in section 6.3. We see that the LLE and Isomap outputs are a bit more difficult to group visually as some clusters are very compact. Again, the LLE outputs would be the easiest to divide into groups, if a coloured separation was not superimposed on the data.

6.4.1 Clustering Validation Indices

The clustering validation indices are derived from the application of K-Means and GK clustering on the two dimensional outputs generated by each algorithm. We can see the results in figure 6.11. A clustering was also performed on the unreduced AML/ALL data set, in order to draw a comparison.

- Dunn Index: The DI index in figure 6.11(a) shows that the clustering of the unreduced data has the highest DI scores. We see, when $r = 8$ and 10 clusters, the score seems to rise and shifts back again for $r = 12$ clusters. In this test the KAS scores were lower than PCA and LLE. The KAS DI scores range from 0.008 to 0.005, whereas the unreduced data set scores are much higher, ending at 0.041 for $r = 12$ clusters. Although, the KAS scores are lower for this particular data set, they are still within a close range to the other algorithms.

- XB Index: The XB clustering index, will have the lowest value for an optimal number of clusters generated by the clustering algorithm. In this case, again, the clustering of the unreduced AML/ALL data had the best XB result beginning at a score of 4.92 for $r = 6$ clusters, and ending at 4.40 for $r = 12$ clusters. When examining the XB scores for the four dimensionality reduction algorithms, the PCA and KAS outputs have the next best XB scores. The KAS scores are initially high and fall back down to 26.02 and 21.33 for $r = 10$ and 12 clusters.

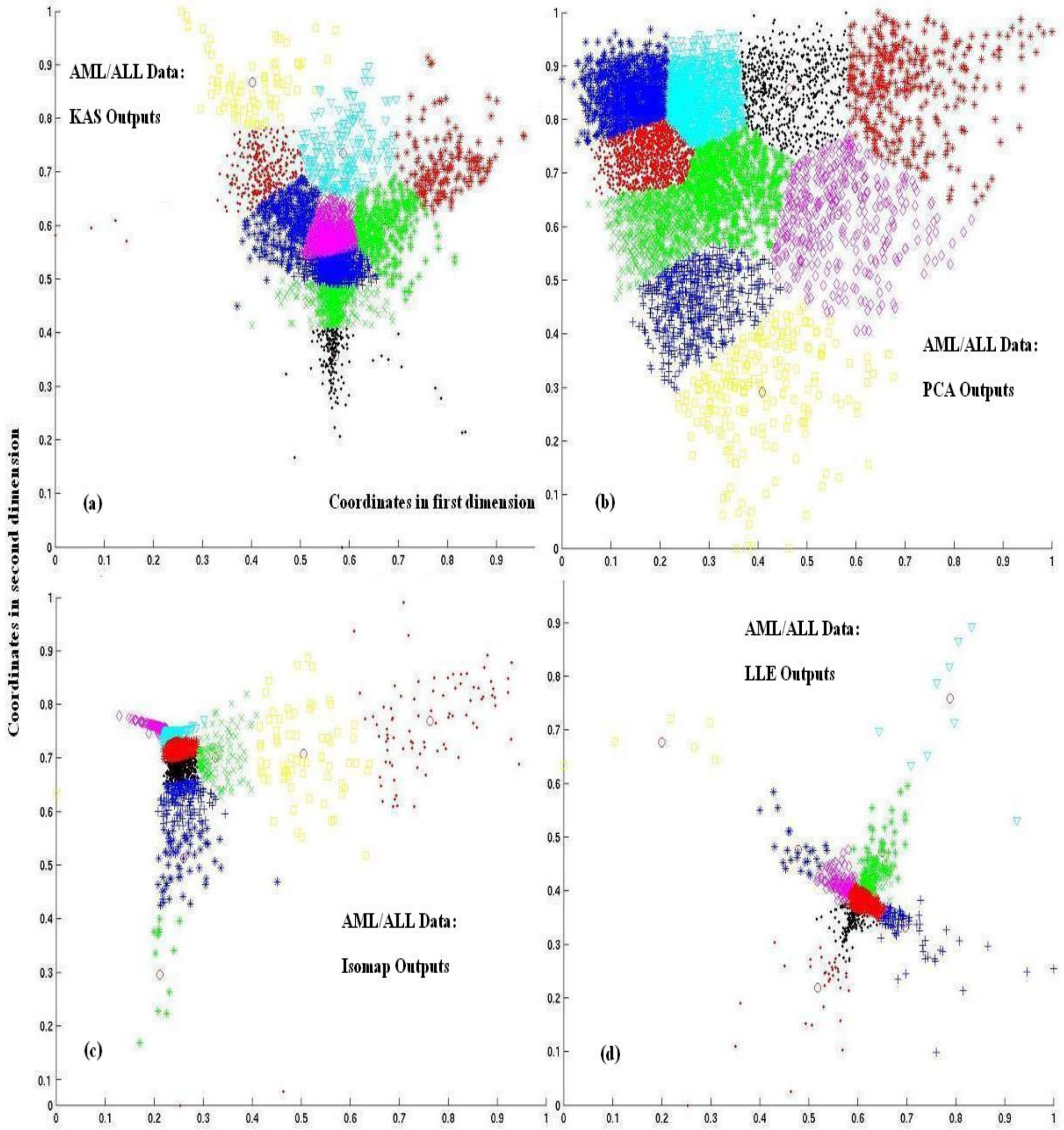


Figure 6.10: Two dimensional outputs for AML/ALL Data. K-Means clustering is applied to the two dimensional data with the number of clusters $r = 10$.

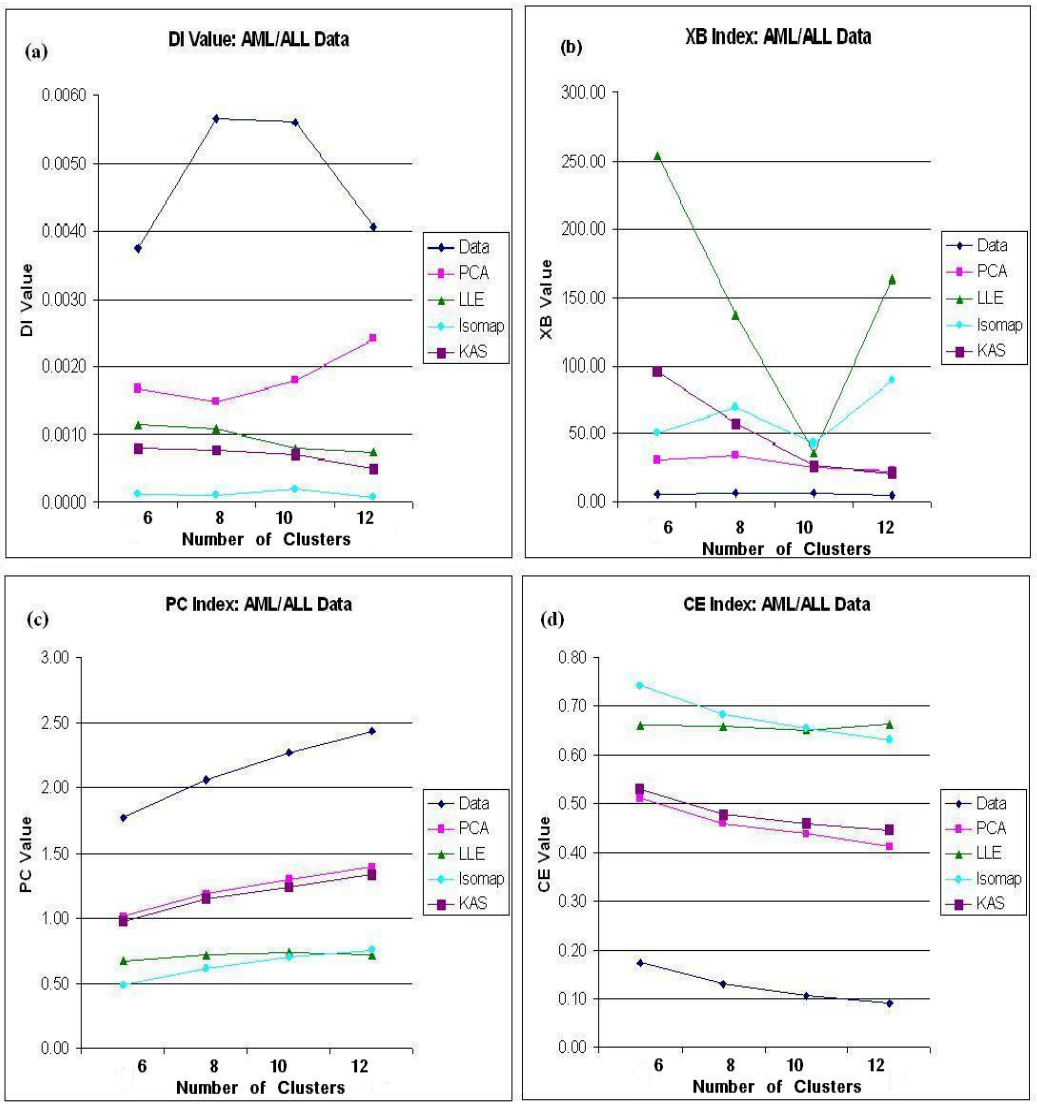


Figure 6.11: Clustering validation indices for K-Means and GK clustering of the AML/ALL data.

Therefore, as the number of clusters increase, the gap between the PCA and KAS scores becomes very little. The Isomap and LLE outputs have XB scores that are not easily interpretable as the scores vary with the number of clusters.

- PC/CE Index: The partition coefficient and classification entropy of a clustering attempts to assess the fuzziness or overlap between the clusters when GK clustering is applied to each output data sets. In figure 6.11(c) and (d) we see the two indices are close to the inverse of one another. In figure 6.11(c), the clustering of the unreduced AML/ALL data attained the highest or best PC score. Similarly, in figure 6.11(b) the clustering of the unreduced AML/ALL data gives the lowest or best CE score. In both cases, the KAS and PCA outputs are close behind in scores. As is similar to the prostate data set, KAS scores are slightly lower than that of the PCA outputs. However, the difference between PCA and KAS scores is very small. The average difference between the two scores for the PC index is 0.045 and 0.02 for the CE index. Again, in comparison to the other algorithms, the Isomap and LLE clusters had the worst PC and CE scores.

6.4.2 Variance in Two Dimensional Embedding

Let us examine figure 6.12 where we see the eigenvalue spectra of the three tested algorithms. KAS retained the highest percentage of variance in the first two dimensions which is approximately 68%. The top three dimensions in the KAS outputs account for approximately 85% of the total variance in the data. This is significant, especially in comparison to the other dimensionality reduction methods. PCA and Isomap generate outputs which require close to six dimensions in order to have similar results to KAS. The Isomap two dimensional embedding retains approximately 25% of total variance, and the PCA two dimensional outputs retain a little over 30% of total variance.

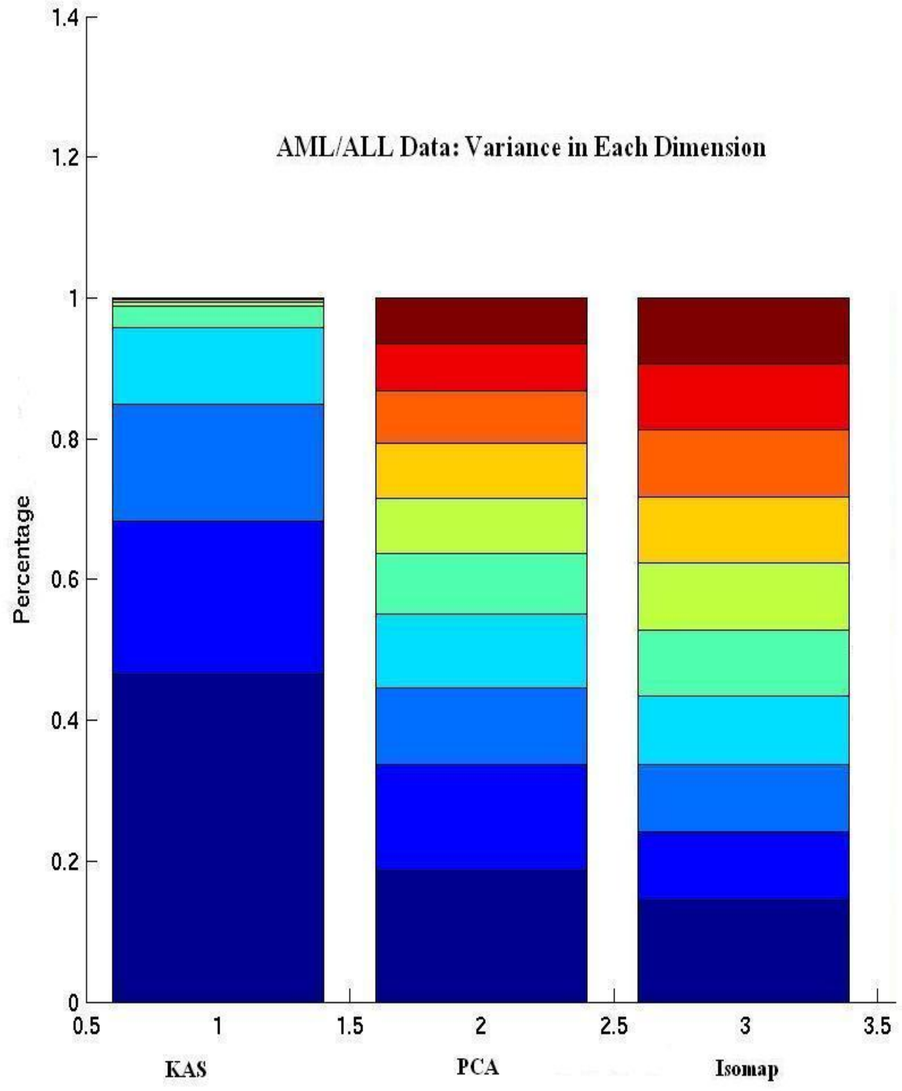


Figure 6.12: Comparison of Variance in each dimension of output generated by dimensionality reduction algorithms. First dimension is dark blue and begins at the bottom of the bar graph. KAS retains highest percentage of variance in two dimensions, which is close to 68% of the total variance contained in the unreduced AML/ALL data set.

6.5 Summary of Results

In this section we summarize the results discussed in sections 6.1 to 6.4.

6.5.1 Two Dimensional Output

The two dimensional depictions of the original high dimensional data sets show great variation across all the dimensionality reduction algorithms. We examined the quality of the two dimensional visual depictions of the data with K-Means clustering applied to the outputs. However, we did not account for additional software tools that could be used to further examine the outputs. Based on the data sets that were tested, we saw the outputs from the PCA algorithm generally span the largest two dimensional area. The KAS algorithm generates two dimensional outputs that are more concentrated in the centre, and generally have fewer genes in each cluster due to the KAS preprocessing steps. K-Means clustering is able to divide the genes clearly for both algorithms. The Isomap outputs have more variation from one dataset to the next and the LLE outputs are the most compact and occupy the least area. The LLE outputs, without a superimposed clustering on the outputs, would be the easiest to separate into distinct groups. This is due to the fact that the LLE outputs are separated across different axes. Although, it is not reasonable to assign a grading to each of the dimensionality reduction algorithms based on visual outputs alone, we can see that ease of visualization is best achieved by the PCA and KAS outputs.

6.5.2 Variance in Two Dimensions

We also examined the eigenvalue spectra of each algorithm in order to assess the two dimensional embedding quality in terms of total variance in the data. Given each experiment with the different data sets, it was clear in every experiment that KAS retained the highest percentage of variance in a two dimensional display of the data. Therefore, it is safe to say that the

KAS outputs have the highest percentage of accuracy in a two dimensional depiction of the data. We saw that the PCA and Isomap algorithms were both far behind in terms of the variance in each output dimension. The one algorithm that we were not able to test for variance, was the LLE algorithm. The LLE algorithm [66] does not allow interpretations of variance in each output dimension, as do the other algorithms tested in this thesis.

6.5.3 Clustering Validation Indices

The clustering validation indices were compared in order to better interpret and assess the quality of the clustering on each of the two dimensional outputs generated from the different algorithms. The clustering validation indices were meant to examine general trends and patterns in the behaviour of the algorithms. An examination of clustering validation indices for the unreduced data sets was used as a basis of comparison. As expected, the unreduced data set had the best scores across the validation indices. We found that the DI and XB results for each clustering showed variation across the data sets. The KAS outputs generated XB scores that were most similar to the unreduced data set in three of the four experiments, and a close second to PCA in the last experiment. The KAS scores for the Dunn Index showed more variation. However, in comparison to the four dimensionality reduction algorithms, KAS scores were either first or second for three of the four experiments. The PC and CE results were fairly straightforward. Again, we saw that the KAS outputs generated PC and CE scores that were closest behind the unreduced data sets in the first and second experiments. However, for the larger data sets, KAS scores were close behind PCA. Again, when examining the clustering validation indices, we were only interested in general behaviour patterns. This is due to the fact that each clustering validation index, in itself, is not an exact measure of clustering quality [3]. We found that clustering applied to the KAS outputs, using either K-Means or GK clustering, faired well in each of the clustering validation indices. KAS did particularly well for the PC, CE and XB indices. This indicates that the

KAS outputs have a low amount of overlap between the clusters, the clusters have low intra-variation and are easily separated and compact. In the chapter that proceeds, we will discuss the conclusions and interpretations of these results.

Chapter 7

Conclusions and Final Remarks

In this section we discuss the results seen in chapter 6 and expand on the behaviour of the dimensionality reduction methods tested in this thesis. We first briefly review the KAS results in section 7.1, followed by a discussion of the other algorithms in section 7.2, and summarize the conclusions in section 7.4.

7.1 KAS Summary

Through the experimentation performed in this thesis, we found that the KAS algorithm proved to be an effective mechanism for the dimensionality reduction of DNA microarray data. The KAS algorithm initially preprocesses the data in order to extract the relevant features of the data and reduce the amount of noise present, by trimming away expression profiles that are not displaying a large amount of variation. Next, the KAS algorithm applies a kernel alignment step that further refines the input data by discarding genes with low alignment values. It is only after both the feature selection and kernel alignment steps that the lSDE algorithm is applied to the data in order to reduce its dimensionality. The KAS results are summarized in subsequent sections.

7.1.1 Inherent Dimensionality

It was seen in sections 6.1 to 6.4, the KAS outputs retained a high percentage of variance in a two dimensional embedding of the data. The PCA and Isomap algorithms were far behind in comparison. The accuracy in the KAS outputs comes from the fact that the *l*SDE dimensionality reduction algorithm applies a more restrictive definition of Isometry than LLE and Isomap. All three of the nonlinear dimensionality reduction algorithms strive to maintain neighbourhoods from the high dimensional space to neighbourhoods in the low dimensional space. However, the SDE and *l*SDE algorithms define a tighter boundary on the idea of a neighbourhood. The nonlinear dimensionality reduction methods were discussed in detail in chapters 3 and 4. The basic idea behind these algorithms is to maintain an input vector from $\mathbf{X}$, and all of its k neighbours, in the low dimensional embedding $\mathbf{Y}$. The LLE algorithm (section 3.4.1) strives to maintain local patches from the high dimensional space to a low dimensional embedding by defining a weight for each neighbour. These weights are subsequently used to form similar neighbourhoods in a low dimensional embedding. The LLE algorithm is limited by the cost function that is used to derive the weights assigned to each neighbour. Therefore, the tightness bound on neighbourhoods and the accuracy of the low dimensional embedding is not guaranteed. The Isomap algorithm also incorporates the idea of Isometry(section 3.4.1), in defining neighbourhoods based on geodesic approximations. The shortest paths to the k nearest neighbours of an input vector are used to define a matrix to which MDS is applied. The idea is that the low dimensional embedding will also have neighbourhoods with similar pairwise distances. Isomap will recover the true dimensionality of data whose intrinsic geometry is that of a convex region of Euclidean space [77]. However, not all data sets can guarantee this property [21]. The SDE and *l*SDE algorithms are able to retain a higher percentage of overall variance due to a tighter definition of what constitutes a neighbourhood. SDE and *l*SDE not only attempt to keep the same neighbours in a low dimensional embedding, but also preserve the respective angles to each neighbour, and distances and angles *between* neighbours from neighbourhoods in $\mathbf{X}$ to neighbourhoods in $\mathbf{Y}$. The SDE and lSDE constraint sets were discussed in section 4.1.3 and 4.1.5. The SDE and lSDE algorithms define the constraints using semidefinite programming and thus, provide guarantees that each neighbourhood constraint will be satisfied in the low dimensional data [86]. Therefore, we can see why the KAS outputs seem to retain a much higher percentage of variance in a low dimensional representation of the original high dimensional data set.

Clustering Indices

The four clustering validation indices that were tested in the results section provide a general measure of validation for the clustering of the KAS two dimensional outputs. The clustering of KAS outputs scored well in terms of the PC(*Partition Coefficient*) and CE(*Classification Entropy*) results. This indicates that the KAS outputs have less overlap between clusters in a fuzzy partition of the data. Therefore, the low dimensional embedding of the genes are more easily separated into distinct groups with less overlap of genes between clusters. This property of KAS is directly related to the kernel alignment step of the algorithm, which is meant to specifically remove genes that are amongst the boundary points between groups.

The XB scores for the KAS algorithm were second to the unreduced data sets in three of the four experiments. It was seen that the unreduced data set had the best overall scores when clustered. The experiment working with the AML/ALL data set had KAS scores that were next best to the unreduced data set and PCA. The XB index attempts to quantify the intra-cluster variation and cluster to cluster separation. Therefore, the KAS scores for the XB index indicate that the KAS outputs can be clustered into separable groups that have low intra-cluster variation. The ability of KAS to form clusters that are distinguishable and detached can, again be attributed to the kernel alignment step of the KAS algorithm.

The KAS scores for the Dunn Index were also good in comparison to the other dimensionality reduction algorithms. For the first two experiments, KAS scores were first and second in comparison to the three other dimensionality reduction methods. Again, the unreduced data set had overall best DI scores. For the last two experiments, KAS scores were second and third in comparison to the three other dimensionality reduction methods. Again, the KAS scores were a little lower for the last experiment using the AML/ALL data. The Dunn Index takes the ratio of the minimum distance between points in differing clusters over the maximum diameter in any cluster. Therefore, the KAS clusters may have a larger diameter or a lower inter-cluster distance. From inspection of the two dimensional outputs we can see that this is true. The KAS outputs sufficiently spread across two dimensions. Consequently, the diameter of each cluster is next largest when compared to the PCA clusters. The KAS clusters also have a relatively low inter-cluster distance. The next section discusses the KAS two dimensional display.

7.1.2 Two Dimensional Display

The KAS and PCA two dimensional displays are the most spread out in two dimensions, across all four experiments. The Isomap two dimensional outputs begin to cramp together as the data sets get larger. The LLE outputs are difficult to assess without additional software visualization tools. However, it is noted in results sections 6.1 to 6.2 that each of the algorithms has positive and negative qualities in a two dimensional display. For example, the KAS outputs allow detection of outliers easier than PCA, since PCA generally depicts data in a dome or circular manner. In the absence of a superimposed clustering on the data, the LLE outputs are the easiest to separate into groups. The fact that the KAS clusters are well spread across two dimensions can be primarily attributed to the maximization constraint in the SDE and lSDE problem definitions. Recall, from chapter 4.1.5, that the selected maximization by Weinberger and Saul [87] is to maximize the variance or distance between the outputs. This effectively results in a well distributed two dimensional embedding that is not achieved by the other nonlinear dimensionality reduction methods.

7.2 PCA, LLE, Isomap Results

In the results sections 6.1 to 6.2 we found the behaviour of the PCA, LLE and Isomap algorithms to be fairly consistent across the four tested data sets. The percentage of variance retained in a two dimensional embedding, for both Isomap and PCA, was quite low. The PCA outputs on average retained approximately 43% of total variance, and the Isomap outputs on average retained approximately 33% of total variance. In chapter 3, we saw that the PCA algorithm is very simple, and does not attempt the study the relationships in the data beyond covariance between input vectors. Also, PCA is not well suited to nonlinear input data (section 3.3). The Isomap algorithm makes an underlying assumption about the input data set, as discussed in section 7.1.1. The assumption of convexity can fail for a number of data sets [21]. Also, as discussed in [21], for small neighbourhoods in $\mathbf{X}$, geodesic distances are equivalent to Euclidean distances. We found in 6.3 and 6.4, that Isomap performance was a little worse for the larger datasets. This is most likely due to the *landmark*-Isomap algorithm that was used for the larger data sets, which compromises accuracy. Both Isomap and PCA had percentage of variance values that were much less than KAS in one to ten dimensions. The PCA algorithm had fairly high scores in terms of the clustering validation indices and slightly outperforms KAS for the larger data sets on the PC and CE indices only. In a two dimensional embedding of the data, the PCA outputs are well distributed and have less overlap between clusters. Therefore, it is intuitive from the two dimensional embeddings, that PCA scores well in the PC and CE indices. LLE scores are typically behind KAS and PCA for the clustering validation indices. LLE performance is typically better when applied to the larger data sets. Isomap scores were generally a little lower for the clustering validation indices. We also found that the performance of the Isomap algorithm was better for the smaller data sets.

7.3 Computational Efficiency

The computational efficiency in terms of execution time was best for PCA, followed by LLE. These algorithms are far faster to run than KAS and Isomap. *Landmark*-Isomap had a faster runtime than the non landmark version. KAS is slower than all of the other dimensionality reduction methods that were tested in this thesis with the exception of the non landmark Isomap algorithm on the larger data sets, which was the slowest. If comparing to LLE alone, KAS is slower by a factor of four times, for the small data sets, and by a factor of eight to nine times for the larger data sets. The bottleneck for the KAS algorithm is the kernel alignment step and the lSDE computation, where the latter takes less time. Part of the lSDE computation includes constraints that restrict the neighbourhood in the low dimensional space to closely resemble neighbourhoods in the high dimensional space. KAS provides guarantees that these neighbourhoods will be maintained in the lower dimensional embedding, where as LLE and Isomap do not have such guarantees. The kernel alignment step in the KAS algorithm is an effective way of assuring KAS outputs are more easily separable into distinguishable groups or clusters. Therefore, although this step may add to the overall computation time, it is an effective addition to dimensionality reduction. KAS does not make any assumptions regarding the input data, as does Isomap. Similar to Isomap and PCA, the eigenvalue spectrum of lSDE reveals the underlying dimensionality of the input data. This is a helpful tool in assessing the reliability of the dimensionality reduction method. The KAS algorithm incorporates steps that attempt to diminish the noise in the data and remove any non differentially expressing genes. Although these two additional steps in the KAS algorithm may be burdensome on the total computation time, they add critical refinements to the dimensionality reduction of microarray data. The other algorithms do not make any prior assessment of the data before dimensionality reduction is performed.

7.4 Conclusions and Application

Our initial results for KAS, as applied to the four tested microarray data sets in this thesis, are quite promising. KAS has proved to be an effective method of dimensionality reduction for microarray data. The lSDE algorithm, employed by KAS, has different properties than algorithms such as PCA, LLE and Isomap. As mentioned in section 7.1.1, many of these differences can be seen as advantages. KAS applies feature selection and kernel alignment in order to effectively reduce the data set before dimensionality reduction is applied. KAS performs well and, in many instances, better than the other dimensionality reduction methods tested in this thesis. KAS can be applied to a high dimensional microarray data set, prior to clustering. We have seen that this will reduce accuracy if compared to the unreduced data, however two dimensional data is far more efficient to cluster than an original data set containing noise, outliers etc. Also, many different clustering methods may be applied to the KAS reduced data, and compared. This would be far too time intensive if working with the unreduced data set. KAS can be used to first reduce microarray data, and subsequently the two dimensional data may be passed on to many different analysis methods, such as support vector machines (SVMs) [57]. SVMs are commonly applied to microarray data for disease classification, however the microarray data must first have a reduced dimensionality. In this case, dimensionality reduction is applied to the rows or genes of the microarray data set while the samples or columns are held constant. In the experimentation for this thesis, we applied KAS to the columns of the microarray data set, while keeping the number of rows constant. However, another important potential application for KAS is to apply dimensionality reduction to the rows of the data set. We also found that KAS was an effective way to better visualize the data in two dimensions. This is a very common approach in assessing the output of microarray experiments. The work presented in this thesis indicates that KAS retains a high level of accuracy in a low dimensional embedding of a high dimensional data set.

7.5 Future Directions

The author of this work would like to explore different facets of this research for possible future direction. Some of these areas are listed below:

- Computational efficiency: An area for futher work is working on the parts of the KAS algorithm that lead to slower computation times. In particular, restructuring the the kernel alignment step of the algorithm may achieve higher running time efficiency.

- KAS for classification: As discussed in section 2.4.1 and 7.4, dimensionality reduction can be applied in two ways to microarray data sets. In this thesis, we explored the application of dimensionality reduction to the columns or samples of the microarray data set, with the number of genes held constant. However, an important direction for future work with KAS, would be applying the dimensionality reduction to the rows of the input data set while keeping the number of samples held constant. This application has a broad range of applications in the study of classification and disease prediction.

- Preprocessing steps of KAS: There is also potential research in examing different preprocessing methods in order to further improve the KAS algorithm

- Other dimensionality reduction methods: An unexplored area of research is the modification of other dimensionality reduction methods to incorporate the KAS preprocessing steps of feature selection and kernel alignment. A comparison between KAS and these modified algorithms would be a very worth while endeavour.

- Testing KAS: In the experimentation for this thesis we tested KAS on four disease related microarray data sets. Further study of KAS applied to timeseries [25] microarray data would be beneficial.

The author of this work would like to thank you for reading through this thesis.

Appendix A

Mathematical Backgound Material

This chapter presents background material for many of the topics discussed in this thesis. References are provided for further detail as needed.

A.1 Definitions

- **Hessian Matrix**
  The Hessian matrix is a square matrix of second order partial derivatives of a scalar-valued function. If we have a vector $x \in R^n$ and a function $f : R^n \rightarrow R$ which has second order partial derivatives, then the Hessian matrix of f is the matrix of second order partial derivatives evaluated at x [79]. The Hessian matrix of a function is denoted by $\mathbf{H}(f)$. Let the vector x be represented as $x = (x_1, x_2, ...x_n)$, where each x_i is a component of x. Therefore, each entry h_{ij} of $\mathbf{H}$ is given by equation (A.1).
  $$h_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j} \tag{A.1}$$

- **Hermitian Matrices**
  A Hermitian matrix is a square matrix that is equal to its own con-

jugate transpose [94]. Therefore, if a square matrix $\mathbf{A}$ is Hermitian, we say that $A = A^H$ and $\mathbf{A}$(i,j)=conjugate($\mathbf{A}$(j,i)). For real matrices, Hermitian and symmetric are equivalent [12].

- **Definite Matrices**
  A real and symmetric matrix $\mathbf{A}$ is positive definite if $x^T\mathbf{A}x > \mathit{0}$ for all nonzero x. The matrix $\mathbf{A}$ is positive *semidefinite* if $x^T\mathbf{A}x \geq \mathit{0}$ for all nonzero x. The matrix $\mathbf{A}$ is considered to be *indefinite* if $x^T\mathbf{A}x > \mathit{0}$ and $x^T\mathbf{A}x < \mathit{0}$ for different nonzero values of x. A real and symmetric matrix $\mathbf{A}$ is positive definite if all of its corresponding eigenvalues are strictly greater than zero. Similarly, a real and symmetric matrix $\mathbf{A}$ is positive *semidefinite* if all of its eigenvalues are greater than or *equal to* zero [88].

- **Manifold**
  A manifold is a topological space which is locally Euclidean [97], meaning that every point has a neighbourhood which resembles Euclidean space and points are separated by Euclidean distances. Examples of manifolds with additional structure include differentiable manifolds and Riemannian manifolds on which distances and angles can be defined [93].

- **Isometry**
  As given in [92] an *isometry* is a mapping of a metric space onto another or onto itself such that the distance between any two points in the original space is the same as the distance between their images in the second space.

- **Translation**
  Motion of an object where the path of every point is a straight line [90].

- **Rotation**
  Motion of an object where the path of every point is a circle or circular arc. A rotation is defined by a point and vector which determine the

axis of rotation. The direction of the vector is the direction of the axis and the magnitude of the vector is the angle of rotation [90].

- **Degrees of Freedom**
  The degrees of freedom (DOF) of an object are the set of independent displacements that specify completely the displaced position of the object [90]. In the two dimensional plane, a rigid object has three degrees of freedom: two translations and one rotation. In three dimensional space, a rigid object has six degrees of freedom: three translations and three rotations [45].

A.2 Mathematical Optimization

A large variety of practical problems involving decision making and quantitative analysis can be perceived and modelled as constrained optimization problems [35]. In this regard, mathematical optimization has become an important tool in many areas such as electronic design automation, engineering, finance, scheduling and other areas involving optimal design problems [9]. Some important definitions related to mathematical optimization are given in A.1. An optimization problem seeks to find the optimal value of a variable $x \in R^n$. The standard form of a mathematical optimization problem is as follows:

$$\begin{aligned} &\textit{minimize: } f_0(x) \\ &\textit{subject to: } f_i(x) \leq b_i, \ \textit{i= 1,...,m.} \end{aligned} \tag{A.2}$$

Where the following problem data is given:

- f_0: $R^n \rightarrow R$ is the objective function,
- b_i are scalars, i = 1,...,m.
- f_i: $R^n \rightarrow R$ are the constraint functions, i = 1,...,m.

The optimal value of $\mathbf{x}$ that satisfies all constraint functions in f_i must also minimize the objective function f_0. Constrained optimization problems may choose to minimize or *maximize* an objective function based on the modelling scenario [13]. Optimization problems can be quite difficult to solve due to often impractical computation times and a solution that is not always guaranteed [8]. However, there are certain classes of optimization problems that can be solved with reasonable computation times and efficiency. This class of problems can be broadly divided into three types.

1. Linear Programming Problems
2. Least Squares Problems
3. Convex Optimization Problems

Linear programming problems and least square problems are a subset of convex optimization problems. Therefore, we continue our discussion on mathematical optimization focussing primarily on convex optimization problems. A detailed review of linear programming and least squares problems can be found in [9].

A.2.1 Convex Optimization Problems

We now define convex sets and properties of convex sets for the purposes of this thesis. If further details are required see [35]. Let $S \subset R^n$. If x_1, $x_2 \in S$ and θ_1, $\theta_2 \in R$, then the line segment joining x_1 and x_2 can be defined as:

$$x = \theta_1 x_1 + \theta_2 x_2, \tag{A.3}$$

where θ_1, $\theta_2 \geq 0$ and $\theta_1 + \theta_2 = 1$. It follows that S is a *convex* set if for any two distinct points in S, the line segment joining these points is also contained completely within S. Therefore, the linear combination of x_1 and x_2 as defined in equation (A.3) must also be contained in S. An example of convex data sets can be seen in figure A.1 and examples of convex functions in figure A.2. In figure A.2, we see that if f_i is a concave function, then $-f_i$ is

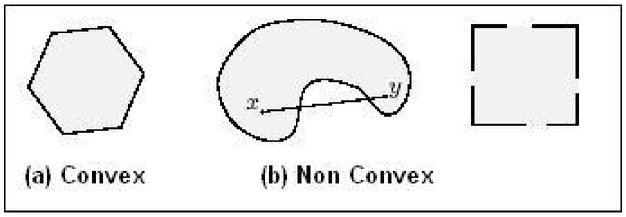


Figure A.1: Difference between convex and non convex data sets diagram from (Figure taken from [35])

a convex function. Convex optimization problems are optimization problems in which the constraint and objective functions from equation (A.2) are all convex [8]. The goal is to minimize or maximize a convex function over a convex set. For example the constraint and objective functions from equation (A.2) must satisfy,

$$f_i(\theta_1 x + \theta_2 y) \leq \theta_1 f_i(x) + \theta_2 f_i(y), \tag{A.4}$$

where x and $y \in R^n$. A general case of equation (A.4) is introduced by replacing the inequality with an equality and simply requiring that θ_1 and θ_2 $\in R$.

$$f_i(\theta_1 x + \theta_2 y) = \theta_1 f_i(x) + \theta_2 f_i(y) \tag{A.5}$$

In this case equation (A.5) is a linear program [8]. Since the inequality of equation (A.4) is replaced by a more restrictive equality in equation (A.5), it follows that linear programming problems are a subset of convex optimization problems. This fact is further discussed in [8]. Also, it can be shown that least squares problems are a subset of convex optimization problems as discussed in [9]. Therefore, the formulation of an optimization problem into a convex form is often desirable. This is due to the fact that convex optimization problems are more general than least square and linear programming problems. Also, an important property of convex optimization problems is the existence of one optimal and global solution [37]. Convex optimization problems are more conducive to larger data sets allowing up to thousands of variables and constraints [37]. Also, there exist many methods

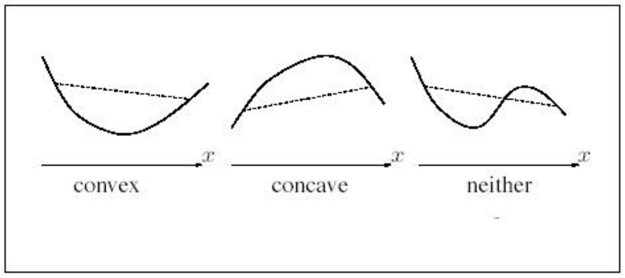


Figure A.2: Convex and non convex functions. (Figure taken from [35])

and tools available to solve convex optimization problems and *semidefinite programming* is one such application.

A.2.2 Semidefinite Programming

In this section the basic semidefinite programming algorithm is discussed. We reviewed a few important matrix definitions in section A.1.

Overview of SDP

Semidefinite programming (SDP) is considered to be the most exciting development in mathematical programming in the past two decades [26]. SDP has applications in such diverse fields as control theory in engineering and operations research. Semidefinite programs are convex optimization problems [80]. Therefore, most of the applications of SDP can usually be solved very efficiently in practice [26]. Let the notation, $\mathbf{F}(\mathrm{x}) \succeq 0$ imply that a matrix $\mathbf{F}(\mathrm{x})$ is positive semidefinite and let the vectors x and $c \in R^m$. Semidefinite programming works with a constraint set defined by $m+1$ symmetric matrices $\mathbf{F_0}...\mathbf{F_m} \in R^{n\times n}$ [80]. The objective is to minimize (*or maximize*) the function $c^T x$ subject to the constraints. As stated above, a semidefinite program is a convex optimization problem because the objective and constraint set are all convex. Therefore, if $\mathbf{F}(\mathrm{x}) \succeq 0$ and $\mathbf{F}(\mathrm{y}) \succeq 0$, then for all

$\theta \in [0,1]$ equation (A.6) follows:

$$F(\theta x + (1-\theta y)) = \theta F(x) + (1-\theta)F(y). \tag{A.6}$$

The standard form of a semidefinite programming problem, as discussed in [9], is given in equation (A.7).

$$\begin{aligned} &\textit{minimize: } c^T x \\ &\textit{subject to: } F(x) \succeq 0 \\ &F(x) = F_0 + \textstyle\sum_{i=1}^{m} F_i x_i. \end{aligned} \tag{A.7}$$

Again, the objective is to minimize $c^T x$ given the restrictions of the problem data. Just as $\mathbf{F}(\mathrm{x}) \geq 0$ signifies that every component of $\mathbf{F}(\mathrm{x})$ must be nonnegative, $\mathbf{F}(\mathrm{x}) \succeq 0$ signifies that each of the n eigenvectors of $\mathbf{F}(\mathrm{x})$ must be nonnegative. In addition to the above SDP problem definition, there exists variations of the exact derivation of the main SDP problem. These can be seen in [80, 26]. Once a problem is formulated into a semidefinite programming problem, there are numerous applications that efficiently solve semidefinite programs. Some of these are Sedumi [74], CSDP [11] and SDPA [29] to name only a few. Semidefinite programming is the basis of the *semidefinite embedding algorithm* discussed in chapter 4. For further details regarding semidefinite programming see [80].

A.3 Distance Metrics

A.3.1 Fractional Distances

Chao and Lihui [70] discuss a fractional metric for determining pair wise distances between inputs in a data set. The fractional distance metric was applied to high dimensional microarray data sets and showed favorable results. In their work, Chao and Lihui also used a contrast metric r, (relative contrast) for any vector x_i in D dimensions [2]. Let $distMax_i$ represent the maximum distance from input x_i to any other input vector. Let $distMin_i$

represent the minimum distance from input x_i to any other input vector. The relative contrast, defined in equation (A.8), is the maximum contrast between *distMax* and *distMin* for x_i to any other input vector.

$$r_i = \frac{distMax_i - distMin_i}{distMin_i} \tag{A.8}$$

It was noted by Aggarwal *et al.* in [2] that the measure of relative contrast r, using Euclidean distances, becomes less reflexive of true relative contrast as the dimensionality of the data set increases. Meaning, r is no longer reflexive of the true contrast in distances between an input x_i and all other inputs. It was shown in [2, 70], a relative contrast based on the fractional distance metrics provided a better approximation of true distances between inputs in high dimensional space. Let $x_{i[k]}$ represent the k^{th} component of i^{th} input vector in D dimensions. The fractional distance metric *dist* from [70], is defined between two inputs x_i and x_j in equation (A.9).

$$dist_{ij} = [\sum_{k=1}^{D} (x_{i[k]} - x_{j[k]})^{\frac{1}{f}}]^f \tag{A.9}$$

Given equation (A.9), having $f = \frac{1}{2}$ results in $dist_{ij}$ being equivalent to a Euclidean distance measure. The findings in [70] suggest using f values between [(0.1)...(0.5)].

A.4 Linear Dimensionality Reduction

A.4.1 Multidimensional Scaling

Multidimensional scaling (MDS), first presented by Cox and Cox [17], is yet another statistical method for finding low dimensional structure from a high dimensional input space. Multidimensional scaling is another nonlinear dimensionality reduction method comparable to PCA. MDS is often applied to the analysis of microarray data [54, 75]. Multidimensional scaling tries to preserve distances between vectors from a high dimensional space to a low dimensional embedding. Typically, MDS computes a distance or dissimilarity matrix from the inputs. The more dissimilarity between two objects in the input space, the greater the expected distance between the two objects in a lower dimensional embedding. In theory, the dissimilarity matrix can be based on any metric that distinguishes the input vectors sufficiently and allows expression through Euclidean distances in the final low dimensional embedding. MDS can be categorized as metric and non-metric MDS. Metric MDS, assumes that the entries in the dissimilarity matrix **D** are Euclidean distances [55]. The non-metric model requires the data to be in the form of ranks [27]. The rankings are used to assign similarity between input vectors. Although there are a wide range of possibilities for the dissimilarity metric [18], a typical form is metric classical MDS (CMDS) [95]. CMDS finds a projection of the input vectors into a low dimensional subspace that best preserves their pairwise squared distances, $|x_i - x_j|^2$, defined in a matrix **D** [87]. The matrix **D** is reformulated into a *(*Gram) or *kernel* matrix **G**, which is defined as a matrix of dot products (see [95] for this derivation). Each element g_{ij} is the dot product between input vectors x_i and x_j as given in equation (A.10).

$$g_{ij} = \langle x_i, x_j \rangle \,. \tag{A.10}$$

Given that the original distance matrix **D** is based on Euclidean distances (metric MDS), the Gram matrix **G** is positive semi-definite. Next, eigendecomposition is performed on the $n \times n$ matrix **G**. Similar to PCA, the outputs **Y** are derived from the eigenvectors having nontrivial eigenvalues. Let λ_k represent the k^{th} eigenvalue, where $k \in [1...d]$. Let **Q** represent the set of orthonormal eigenvectors of **G**. Assume the eigenvalues and corresponding eigenvectors are maintained in descending order based on the eigenvalue spectra. The k^{th} embedding coordinate for each output y_i can be derived as follows [55]:

$$y_{ik} = \sqrt{\lambda_k} Q_{ik}. \tag{A.11}$$

MDS is applied to microarray data for analysis in many current software applications such as [38, 83, 67] to name a few. Also, in [75] Taguchi and

Oono applied a novel nonmetric MDS approach to microarray data attaining promising results. In addition, MDS can be utilized for qualitative two or three dimensional displays rather than serious quantitative analysis [54]. Dimensionality reduction techniques such as PCA and MDS are quite versatile in applications such as clustering microarray data sets. Frequently, DNA microarray data undergoes clustering analysis whereby the genes are divided into groups or clusters with like expression patterns. Various clustering algorithms, such as k-means clustering, require specifying the number of partitions in advance [59]. Often this is difficult to estimate and a tool such as PCA or MDS is required to visualize the data and estimate the number of partitions in two or three dimensions.

A.5 Nonlinear Dimensionality Reduction

A.5.1 Laplacian Eigenmaps

Laplacian Eigenmaps is another unsupervised manifold learning algorithm that reduces the dimensionality of large data sets. First introduced to manifold learning in 2001 by Belkin and Niyogi [49], the algorithm first takes the *Laplacian* of a graph of input data in order to construct a Laplacian matrix. This approximates the *Laplace-Beltrami* operator defined for the input data. The Laplacian matrix is solved using eigenanalysis in order to construct a d dimensional embedding of the original input data. It was stated by the authors of [49, 84], that the Laplacian Eigenmaps algorithm is insensitive to outliers and noise, yet is able to highlight the natural clusters and groupings of the data in low dimensional space. The algorithm is also quite simple and not extremely computationally intensive, with local computations and one sparse eigenvalue problem that is given in step(4):

1. Construct a weighted graph with nodes representing every input vector x_i. Edges occur between neighbouring nodes. Neighbours are based on

one of the following:

   (a) $\|x_i - x_j\|^2 < \epsilon$

   (b) k nearest neighbours

2. Construct a symmetric weights matrix, **W**. Assign w_{ij} to be the weight of an edge between two nodes based on one of the following:

   (a) Heat function:

$$w_{ij} = e^{-\frac{\|x_i - x_j\|^2}{t}} \ \textit{where } t \in R \tag{A.12}$$

   (b) w_{ij} =1 if x_i and x_j are neighbours, else $w_{ij} = 0$

3. Let **D** represent a diagonal weight matrix from the column or row sums of **W**. Let **L**= **D** -**W**, a Laplacian matrix which is symmetric and positive semidefinite.

4. Solve the eigenvalue problem in equation (A.13).

$$\mathbf{L}y = \lambda \mathbf{D}y. \tag{A.13}$$

   Obtain the d highest order eigenvectors, based on the eigenvalues with the largest percentage of variance, and construct a low dimensional embedding directly from the eigenvectors.

Laplacian Eigenmaps have been applied to vision, dictionary, and speech data in [49]. The results indicate that Laplacian Eigenmaps are able to illustrate relationships in the data far better than traditional PCA. The Laplacian Eigenmaps algorithm has been formulated as a semi-supervised learning approach as illustrated in [51]. Laplacian Eigenmaps for embedding and clustering was assessed in [50], where a linear extension to Laplacian Eigenmaps was also proposed. At the time of this thesis, to the best of our knowledge, there is no existing publication reviewing the application of Laplacian Eigenmaps to DNA microarray data.

A.5.2 Hessian Eigenmaps

Hessian Eigenmaps, or Hessian based local linear embedding, has a theoretical bases similar to Laplacian Eigenmaps and LLE. Introduced by Donoho and Grimes in [21], Hessian Eigenmaps replaces the Laplacian with a quadratic form based on the Hessian (see section A.1 for further details on Hessian matrices). Hessian Eigenmaps also follows the basic steps of the LLE algorithm discussed in section 3.4.1. In principle, Hessian Eigenmaps is a dimensionality reduction method meant to preserve isometry when mapping from a high dimensional manifold to a low dimensional submanifold, without the restriction that an underlying submanifold must be a convex subset of the original. A general overview of the Hessian Eigenmaps algorithm, as described in further detail in [21], follows:

1. Identify the k nearest neighbours of each input vector x_i. For each neighbourhood, construct a $k \times D$ matrix $\mathbf{M^i}$. Let μ_i represent the average of all k neighbours of x_i. The rows of $\mathbf{M^i}$ consist of the k neighbours of x_i. However, each row or neighbour x_j is re-centered based on μ_i.
$$x_j \leftarrow x_j - \mu_i. \tag{A.14}$$

2. Define a smooth function f, mapping each neighbourhood matrix $\mathbf{M^i}$ to R (f: $\mathbf{M^i} \rightarrow$ R). Let the Hessian matrix of f be represented by $\mathbf{H}$. Construct a least square approximation of $\mathbf{H}$. The least square approximation matrix is denoted by $\dot{\mathbf{H}}$.

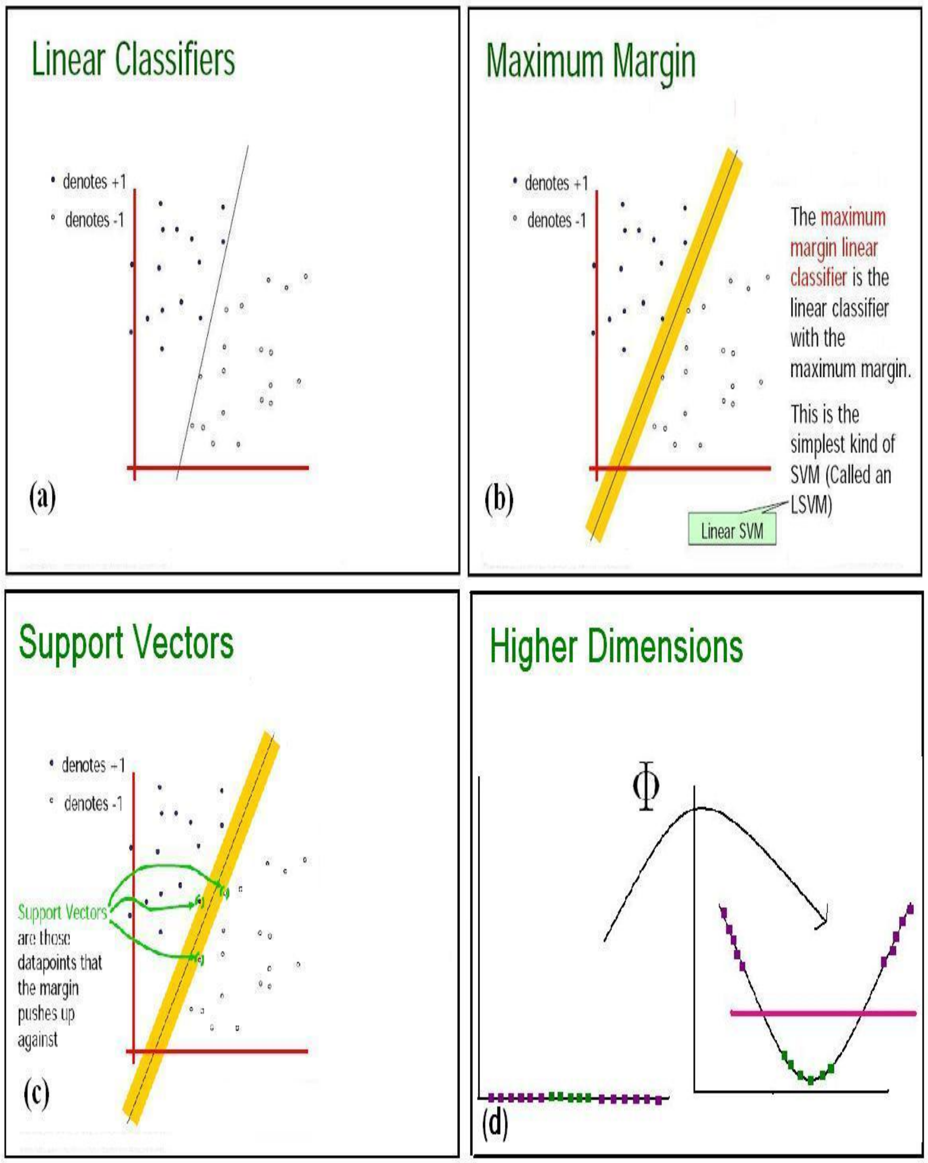
3. Derive the quadratic form $\hat{\mathbf{H}}$ of the Hessian approximation matrix $\dot{\mathbf{H}}$. Perform eigenanalysis on $\hat{\mathbf{H}}$ as given in equation (A.15).
$$\lambda y = \hat{\mathbf{H}} y. \tag{A.15}$$

4. Derive the low dimensional embedding in a similar manner to other nonlinear dimensionality algorithms. The output vectors $\mathbf{Y}$ are equal to the eigenvectors of $\hat{\mathbf{H}}$ that have the largest percentage of variance.

The possibility to integrate noise aware methodologies and statistical modelling in Hessian Eigenmaps is discussed in [21]. HLLE was tested on wireless sensor networks in [53] and, in certain classes of problems, demonstrated superior results over other nonlinear dimensionality reduction methods. Experiments on LLE, Isomap, and Laplacian Eigenmaps in [5] showed that extending these algorithms to formulate an out-of-sample prediction function, such as that given in equation (3.9), is possible and quite effective. Similar extensions can be applied to Hessian Eigenmaps using the Nystrom formula [4, 5]. At the time of this writing, as is the case with Laplacian Eigenmaps, we have not come across any published work on the application of Hessian Eigenmaps to DNA microarray data.

A.6 Support Vector Machines

A support vector machine (SVM) is a supervised learning method that is used for data classification and regression analysis. For the purposes of data classification, support vector machines separate the data into one of two classes. For example, if the data set of interest is two dimensional, as in figure A.3(a), the goal is to separate the data into two groups and any linear classifier can do this to a reasonable extent. The linear classifier that separates the two classes by a maximum margin, is called the maximum margin classifier as given in figure A.3(b). The aim of support vector machines is to classify the data with a maximum margin between the two classes. Therefore, a linear support vector machine for two dimensional data is also a maximum margin classifier. We see that *support vectors* as seen in figure A.3(c) are the points, or vectors in higher dimensional data, that are bordering on the margin. In the case of multidimensional data, a *hyperplane* is desired that separates the data. The optimal hyperplane is known as the maximum margin hyperplane and the support vectors are the vectors that are bordering on the hyperplane from either of the two groups. However, often data is not easily separable, and linear classification is not possible in the original dimension. A two di-



The maximum margin linear classifier is the linear classifier with the maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Support Vectors are those datapoints that the margin pushes up against

Figure A.3: Introductory slides on support vector machines. (Figures modified from [47])

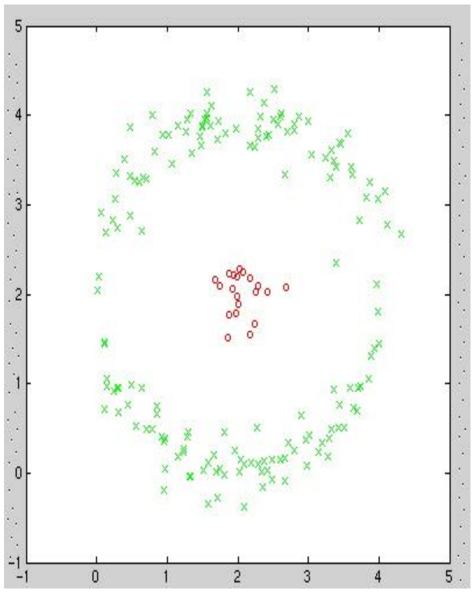


Figure A.4: An example of data that is inseparable by a linear classifier. This figure was taken from [24]

mensional example of this can be seen in figure A.4. Similarly, when dealing with multidimensional data, there may not be a guaranteed hyperplane that definitively classifies every vector into one of two groups. In this case, the inseparable data is mapped to higher dimensions called a *feature space*, where the data becomes separable. SVMs employ a popular method used in kernel based learning algorithms, known as the *kernel trick* and this is discussed in the next section.

A.6.1 Kernel Trick

In machine based learning algorithms, often the data is inseparable in its original form. VC (Vapnik-Chervonenkis) theory [82] explains that often mappings which take us into a space that is of higher dimension than that of the input space, results in improved classification power [24]. The kernel trick is a method of projecting the data into a higher dimensional space, where the data becomes separable by an ordinary linear classification method.

Therefore, in the higher dimensional space linear classification is equivalent to non-linear classification in the original space. An simple example of this can be seen in figure A.3(d). The problem of mapping into higher dimensions is that this can seriously increase computation time if we have to explicitly compute the mappings for every input vector [24]. Let a mapping function be represented by $\Phi : R^n \rightarrow R^m$, where $n < m$. Instead of using the mapping function explicitly we work with the dot products between input vectors mapped by Φ. Let K represent a kernel function which is equivalent to the dot product between $\Phi(x_i)$ and $\Phi(x_j)$ as given in equation (A.16). Therefore, given such a kernel function, we can compute $K(x_i, x_j)$ and never have to compute $\Phi(x_i)$ or $\Phi(x_i)$ explicity.

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle \tag{A.16}$$

Mercer's theorem [41] states that for some Φ, a symmetric function $K(x_i, x_j)$ can be expressed as the inner product in equation (A.16), if and only if $K(x_i, x_j)$ is a positive semidefinite matrix (see A.1 for definitions). Therefore, if we can appropriately choose $K(x_i, x_j)$, then subsequent processing can be carried out using dot products alone. The function Φ is never computed, as it is not needed if one works only with the dot products. By employing the kernel trick, SVM's first project data into a higher dimensional feature space. The mapping function, Φ never needs to be calculated, as SVM's are able to work exclusively with the dot product given in equation A.16. This allows support vector machines to find a maximum margin hyperplane for data that is initially inseparable. Kernels are used in many areas to ease the transformation of data into higher dimensions. Examples of kernel functions are given in [47, 24, 41]. See [47] for further information on support vector machines.

Bibliography

[1] Alon A, Barkai N, Notterman D.A., Gish K., Ybarra S., Mack D., and Levine A.J. Expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci.*, 1:6745–6750, 1999.

[2] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional space. *Lecture Notes in Computer Science*, 1973, 2001.

[3] Balazs Balasko, Janos Abonyi, and Balazs Feil. Fuzzy clustering and data analysis toolbox for use with matlab.

[4] Yoshua Bengio, Olivier Delalleau, Nicolas Le Roux, Jean-Francois Paiement, Pascal Vincent, and Marie Ouimet. Learning eigenfunctions links spectral embedding and kernel pca. *Neural Computation*, 16(10):2197–2219, 2004.

[5] Yoshua Bengio, Olivier Delalleau, Nicolas Le Roux, Jean-Francois Paiement, Pascal Vincent, and Marie Ouimet. Spectral dimensionality reduction. Technical Report 2004s-27, CIRANO, May 2004.

[6] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[7] Joshua E. Blumenstock. Mining clusters for knowledge: Finding algorithm-independent groups in microarray data, 2003.

[8] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[9] Stephen Boyd and Lieven Vandenberghe. Convex optimization, semidefinite programing and recent applications. In *COLT*, Banff, Canada, July 2004.

[10] Kristin Branson. Global geometric frameworks for nonlinear dimensionality reduction: The isomap and locally linear embedding algorithms, 2002.

[11] B. Brochers. A c library for semidefinite programming. *Optimization Methods and Software*, 11:613–623, 1999.

[12] Mike Brookes. The matrix reference manual. http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/property.html, 2005.

[13] Chistopher Burges. Some mathematical tools for machine learning. Technical report, Microsoft Research, Max Planck Institute, T*ü*bingen, 2003.

[14] Anthony Carpi and Dr. G. Weaver. The cell. http://web.jjay.cuny.edu/∼acarpi/NSC/13-cells.htm, 1999.

[15] Bradley Coe and Christine Antler. Spot your genes - an overview of the microarray. http://www.scq.ubc.ca/?p=272, 2006.

[16] Affymetrix Corporation. http://www.affymetrix.com.

[17] M.AA Cox and T.F. Cox. Local minima in nonmetric multidimensional scaling. Technical report.

[18] M.AA Cox and T.F. Cox. *Multidimensional Scaling*. CRC/Chapman and Hall, 2001.

[19] Nello Cristianini, John Shawe-Taylor, Andr*é* Elisseeff, and Jaz Kandola. On kernel target alignment. 2001.

[20] Vin de Silva and Joshua B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. citeseer.ist.psu.edu/desilva03global.html, 2003.

[21] David L. Donoho and Carrie Grimes. Hessian eigenmaps: new locally linear embedding techniques for highdimensional data. Technical report, Stanford University, 2003.

[22] Sandrine Dudoit, Robert Gentleman, Rafael Irizarry, and Yee HwaYang. Dna microarray data oligonucleotide arrays. Bioconductor Short Course, 2003.

[23] Edsger Wybe Dijkstra. A note on two problems in connection with graphs. *Numerical Mathematics*, 1(5):269–271, 1959.

[24] Ian Fasel. Kernel pca : Schölkopf smola and müller: Nonlinear component analysis as a kernel eigenvalue problem. *Vision and Learning*, 2001.

[25] Kurt Fellenberg, Nicole C. Hauser, Benedikt Brors, Albert Neutzner, Jörg D. Hoheisel, and Martin Vingron. Correspondence analysis applied to microarray data. *PNAS*, 98(19):10781–10786, 2001.

[26] Robert M. Freund. *Introduction to Semidefinite Programming (SDP)*, 2004.

[27] Tobias Friedrich. Nonlinear dimensionality reduction with locally linear embedding and isomap. Master's thesis, Department of Computer Science, The University of Sheffield, 2002.

[28] Li-Min Fu and Enzo Medico. Fmc, a fuzzy map clustering algorithm for microarray data analysis. *BITS: Bioinformatics Italian Society Meeting*, 2004.

[29] K. Fujisawa, M. Fukuda, M. Kojima, and K. Nakata. Numerical evaluation of sdpa (semidefinite programming algorithm. Technical report, 1997.

[30] Robert Glaubius, Motoi Namihira, and William D. Smart. Speeding up reinforcement learning using manifold representations: Preliminary results. 2005.

[31] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[32] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17:2/3:107–145, 2001.

[33] A. Ben Hamza and Hamid Krim. Geodesic matching of triangulated surfaces. http://users.encs.concordia.ca/∼hamza/TIPdouble.pdf.

[34] T. Hastie, R. Tibshirani, and J. Friedman. The elements of statistical learning. Springer-Verlag, 2001.

[35] Haitham Hindi. A tutorial on convex optimization. Technical report, Palo Alto Research Centre, Palo Alto, California, 2002.

[36] Jaakko Hollmén. Principal component analysis. http://www.cis.hut.fi/∼jhollmen/dippa/node30.html, 1996.

[37] Frontline Systems Incorporated. http://www.solver.com/probconvex.htm.

[38] Partek Incorporated. Software for dna microarray data analysis:. http://www.partek.com.

[39] Pearson Education Incorporated. Dna structure. http://academic.brooklyn.cuny.edu/biology/ bio4fv/page/molecular%20biology/dna-structure.html.

[40] Jens Nilsson and Thoas Fioretos and Mattias Höglund and Magnus Fontes. Approximate geodesic distances reveal biologically relevant structures in microarray data. *Bioinformatics*, 20(6):874–880, 2004.

[41] Lecturer Michael Jordan. Advanced topics in learning decision making-the kernel trick. http://www.cs.berkeley.edu/∼jordan/courses/281A-fall04/.

[42] Fumiaki Katagiri and Jane Glazebrook. Local context finder (lcf) reveals multidimensional relationships among mrna expression profiles of arabidopsis responding to pathogen infection. *PNAS*, 100(19):10842–10847, 2003.

[43] Zheng Li. Kernel based nonlinear feature extraction in bioinformatics. Technical report, 2005.

[44] Henry Horng-Shing Lu and Han-Ming Wu. On visualization, screening, and classification of cell cycle-regulated genes in yeast. *Genome Informatics*, 14:344–345, 2003.

[45] Kirk Martini. http://urban.arch.virginia.edu/∼km6e/references/glossary/struc-glossary.html, 1997.

[46] Klaus-Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–200, 2001.

[47] Andrew W. Moore. Support vector machines. Technical report, 2001.

[48] Partha Niyogi and Mikhail Belkin. An introduction to manifold methods. University of Chicago.

[49] Partha Niyogi and Mikhail Belkin. Laplacian eigenmaps for dimensionality reduction and data representation. Technical report.

[50] Partha Niyogi and Mikhail Belkin. Laplacian eigenmaps and spectral techniques for embedding and clustering. citeseer.ist.psu.edu/belkin01laplacian.html, 2002.

[51] Partha Niyogi and Mikhail Belkin. Semi-supervised learning on riemannian manifolds. *Machine Learning*, 56:209–239, 2004.

[52] D. Botstein O. Alter, P. Brown. Singular value decomposition for genome-wide expression data processing and modeling. *Proc. Natl. Acad. Sci.*, 97:10101–10106, 2000.

[53] Neal Patwari and Alfred O. Hero III. Manifold learning algorithms for localization in wireless sensor networks. University of Michigan, 2003.

[54] Carsten Peterson and Markus Ringnér. Analyzing tumor gene expression profiles. *Artificial Intelligence in Medicine*, 28:59–74, 2003.

[55] John C. Platt. Fastmap, metricmap, and landmark mds are all nystom algorithms. Technical report, Microsoft Research, 1 Microsoft Way.

[56] John C. Platt. Fast embedding of sparse similarity graphs. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[57] Nathalie Pochet, Frank De Smet, Johan A.K. Suykens, and Bart L.R. De Moor. Systematic benchmarking of microarray data classification: Assessing the role of nonlinearity and dimensionality reduction. *Bioinformatics*, 17:3185–95, 2004.

[58] Spellman P.T., Sherlock G., Zhang M.Q., Iyer V.R., Anders K, Eisen M.B., Brown P.O., Botstein D., and Futcher B. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization, 1998.

[59] John Quackenbush. Computational analysis of microarray data. *Neural Computing and Applications*, 2:418–427, 2001.

[60] M. Rattray, N. Morrison, D. Hoyle, and A. Brass. DNA microarray normalisation and pca. University of Manchester, 2001.

[61] S. Raychaudhuri, J.M. Stuart, and R.B. Altman. Principal components analysis to summarize microarray experiments: application to sporulation time series. 2000.

[62] Robert W. Floyd. Algorithm 97 (shortest path). *Communications of the ACM*, 5(6):345, 1962.

[63] R. Rosipal, M. Girolami, L. Trejo, and A. Cichocki. Kernel-pca for feature extraction and de-noising in non linear regression. *Neural Computing and Applications*, pages 10:231–243, 2001.

[64] Sam T. Roweis and Lawrence K. Saul. A introduction to local linear embedding. Technical report, Gatsby Computational Neuroscience Unit, AT&T Labs Research.

[65] Sam T. Roweis and Lawrence K. Saul. Pseudocode for lle algorithm. http://www.cs.toronto.edu/∼roweis/lle/algorithm.html.

[66] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by local linear embedding. *Science*, 290:2323–2326, 2000.

[67] Lao H Saal, Carl Troein, Johan Vallon-Christersson, Sofia Gruvberger, Ake Borg, and Carsten Peterson. Bioarray software environment (base): a platform for comprehensive management and analysis of microarray data. http://genomebiology.com/2002/3/8/software/0003.1.

[68] Jessica Shah. A review of dna microarray data analysis. Technical report, Stanford University.

[69] R. Shealy, S. Clough, R. Philip, A. Khanna, and L. Vodkin. Analysis of microarray data. Technical report.

[70] Chao Shi and Lihui Chen. Feature dimension reduction for microarray data analysis using locally linear embedding. In *APBC*, pages 211–217, 2005.

[71] Dinesh Singh, Phillip G. Febbo, Kenneth Ross, Donald G. Jackson, Judith Manola, Christine Ladd, Pablo Tamayo, Andrew A. Renshaw, Anthony V. D'Amico, Jerome P. Richie, Eric S. Lander, Massimo Loda, Philip W. Kantoff, Todd R. Golub, and William R. Sellers. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1(2):203–209, 2002.

[72] Lindsay Smith. A tutorial on principal components analysis. http://csnet.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf, 2002.

[73] Thomas Strohman. Using kernel alignment to pick support vectors. http://csel.cs.colorado.edu/∼strohman/KA_SVM_paper.pdf, 2004.

[74] J.F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653, 1999. Special issue on Interior Point Methods (CD supplement with software).

[75] Yh. Taguchi and Y. Oono. C. elegans microarray data seen through a novel nonmetric multidimensional scaling method. Chuo University UIUC.

[76] Yongxi Tan, Leming Shi, Weida Tong, and Charles Wang. Multi-class cancer classification by total principal component regression (tpcr) using microarray gene expression data. *Nucleic Acids Research*, 33, 2005.

[77] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[78] Burak Turhan. Nonlinear dimensionality reduction methods for pattern recognition. Master's thesis, 2004.

[79] Cl*á*udio Valente and Thomas Foregger. http://planetmath.org/encyclopedia/HessianMatrix.html.

[80] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. Technical report, 1994.

[81] van't Veer L.J., Dai H., van de Vijver M.J., He Y.D., Hart A.A.M., Mao M., Peterse H.L., van der Kooy K., Marton M.J., Witteveen A.T., Schreiber G.J., Kerkhoven R.M., Roberts C., Linsley P.S., Bernards R., and Friend S.H. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415:530–536, 2002.

[82] Vladimir Naumovich Vapnik. *The Nature of Statistical Learning Theory.* Second edition, 1998.

[83] VisuMap. http://www.visumap.net/index.aspx?p=Home.

[84] Lisa Wainer and Dr.Pontil. Laplacian eigenmaps for dimensionality reduction and data representation. University College London.

[85] Zuyi Wang, Yue Wang, Jianping Lu, Sun-Yuan Kung, Junying Zhang, Richard Lee, Jianhua Xuan, Javed Khan, and Robert Clarke. Discriminatory mining of gene expression microarray data. *J. VLSI Signal Process. Syst.*, 35(3):255–272, 2003.

[86] Kilian Weinberger, Benjamin D. Packer, and Lawrence K. Saul. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. *AISTATS*, 2005.

[87] Kilian Weinberger and Lawrence K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *Computer Vision and Pattern Recognition*, 2004.

[88] Eric W. Weisstein. Positive definite matrix. http://mathworld.wolfram.com/PositiveDefiniteMatrix.html.

[89] Wikipedia. Covariance — wikipedia, the free encyclopedia. www.wikipedia.org, 2006. [Online; accessed 24-August-2006].

[90] Wikipedia. Degrees of freedom (engineering) — wikipedia, the free encyclopedia. www.wikipedia.org, 2006. [Online; accessed 23-August-2006].

[91] Wikipedia. Distance (graph theory) — wikipedia, the free encyclopedia. www.wikipedia.org, 2006. [Online; accessed 22-July-2006].

[92] Wikipedia. Isometry — wikipedia, the free encyclopedia. www.wikipedia.org, 2006. [Online; accessed 24-August-2006].

[93] Wikipedia. Manifold — wikipedia, the free encyclopedia. www.wikipedia.org, 2006. [Online; accessed 19-August-2006].

[94] Wikipedia. Self-adjoint operator — wikipedia, the free encyclopedia. www.wikipedia.org, 2006. [Online; accessed 18-August-2006].

[95] Forrest W. Young. Multidimensional scaling. *Kotz-Johnson (Ed.) Encyclopedia of Statistical Sciences*, 5, 1985.

[96] Chaolin Zhang, Yanda Li, and Xuegong Zhang. Gmap: extracting and interactively visualizing nonlinear relationships of genes from expression. *Proceedings of RECOMB04*, pages 228–229, 2004.

[97] J. Zhang, S. Li, and J. Wang. Nearest manifold approach for face recognition. *The 6th IEEE International Conference on Automatic Face and Gesture Recognition*, 2004.