A study of dimensionality reduction as subspace data embedding linked by multidimensional scaling.

Wang, Wanqiu.

2005

Wang, W. (2005). A study of dimensionality reduction as subspace data embedding linked by multidimensional scaling. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/3685>

<https://doi.org/10.32657/10356/3685>

Nanyang Technological University

Downloaded on 26 Apr 2023 18:54:55 SGT

A Study of Dimensionality Reduction as Subspace Data Embedding Linked by Multidimensional Scaling

WANG WANQIU

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Master of Engineering

2005

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

23 June 2005
.....

Date

.....

WANG WANQIU

*To my parents and Xzufeng,
for their encouragement and love.*

Acknowledgments

I am grateful to Associate Professor Kap Luk Chan for being my supervisor and for his support and inspiration. This work has not been possible without his guidance and encouragement. I am also thankful to Dr. Zhihua Zhang, whose enthusiasm towards better research motivated me deeply. He helped me broaden my mathematical knowledge, and encouraged me towards further study. If one day, I choose research as my career, he is the one that helped me to choose that path.

I would like to thank Dr. Lei Wang, Dr. Chibiao Chen, Dr. Minchun Liu, Mr. Yiming Wu, and many others for their valuable and interesting discussions.

Finally, I would like to thank my parents and brother for their support since I was born.

Summary

Some state-of-the-art dimensionality reduction techniques are reviewed and investigated in this thesis. Dimensionality reduction techniques can be categorized into two serving different purposes. The first category is to mitigate the computational load and to address the *Curse-of-Dimensionality*, and the second category is to model the data spread or manifold. ISOMAP and LLE techniques are developed for the second purpose, and both of them are embedding techniques. By means of embedding, some data points in a higher dimensional space can be mapped into a lower dimensional space, provided that the pairwise distances are kept unchanged or within a small tolerant range. Some conventional category one techniques, such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), are developed from variance analysis, but they also can be interpreted as embedding techniques through links to the metric Multidimensional Scaling (MDS).

PCA and LDA are both commonly used dimensionality reduction techniques. However, it is not obvious that there is a connection between these two except that both are techniques based on analysis of variance. Indeed, these two have been frequently put into contrasting roles, such that PCA preserves the variance, and LDA obtains the discriminant dimensions. Actually, both of them can be interpreted as embedding techniques through the metric MDS; as such, a natural link can be established between these two methods. It is observed that they exhibit similarities in some respects. The only two differences between PCA and LDA are, first, they select different representatives in embedding. Second, PCA

embeds the Euclidean distances, while LDA embeds Mahalanobis distances. From the observation, a framework can be developed to accommodate both methods.

Recently, kernel method became popular due to its success in machine learning and pattern recognition. Some conventional dimensionality reduction techniques like PCA and LDA have also been generalized into the kernel (feature) space. Indeed, the kernel PCA can also be interpreted as a special case of metric MDS, of course, with a special distance metric generated from the kernel. A more generalized distance metric can be obtained by utilizing the convolution kernel, where we can even combine the data vector and label information into a single metric. Through an embedding process, discriminant dimensions can be extracted.

PCA and LDA are based on linear models for feature extraction. In other words, these methods are only suitable if the data samples are drawn from a linear model or perturbations from such a model. For some special data manifolds, such as a spiral, it is impossible to extract the true structures using these two techniques. The embedding techniques, however, can be generalized to solve such problems. ISOMAP, as a generalized metric MDS technique, can find the true structure successfully. The key point for the success of ISOMAP is that the distance metrics are set to be different for neighboring and non-neighboring data points. As such, the locally linear and globally non-linear structure can be preserved in the embedding. In contrast to ISOMAP's explicit way to differentiate the distance metrics for neighboring and non-neighboring data points, LLE uses two optimization steps in embedding, where the distance metrics for neighboring and non-neighboring data points are implicitly differentiated.

Through the discussion of the links between some above-mentioned techniques, an embedding based framework is developed in this thesis. This framework first selects representatives from the training samples if needed. Then it calculates all the pairwise distances and forms a squared distance matrix D^2 by filling all the pairwise distances. Finally, metric MDS is applied to obtain a possibly lower dimensional space which can accommodate all the training samples while preserving the pairwise distances as much as possible. Under this framework, the selection of the representatives and the tuning of the distance metric may derive different subspaces serving different purposes. For example, both PCA and LDA can be unified under one roof by this framework to derive a generative subspace and a discriminative subspace, respectively.

Face recognition, as a typical pattern recognition problem, is chosen to evaluate those dimensionality reduction techniques belonging to the first category and their possible embedding versions as feature extraction techniques. Through the experiments, we can see that the embedded versions can give similar performances to their conventional counterparts. Besides, under this framework, carefully tuned distance metrics can be used but conventional techniques are not easy to be generalized with such capabilities. In the experiments, an ad-hoc convolutional kernel generated distance metric is applied. In addition, it is shown that for both AT&T and Yale databases, the class means are sufficient for training the face space.

Contents

Acknowledgements	i
Summary	ii
List of Figures	ix
List of Tables	xi
1 Introduction	1
1.1 Motivation of Dimensionality Reduction	1
1.2 A Brief Review of Some Dimensionality Reduction Techniques .	3
1.3 Objective and Major Contributions of This Thesis	7
1.4 Organization of This Thesis	9
2 Multidimensional Scaling	11
2.1 Introduction	11
2.2 Types of MDS	12
2.2.1 Metric scaling.	12
2.2.2 Nonmetric Scaling	13

2.2.3	Three-way Scaling	14
2.3	Embedding Algorithms for Metric MDS	14
2.3.1	Loss Functions	15
2.3.1.1	STRESS	15
2.3.1.2	SSTRESS	16
2.3.1.3	STRAIN	17
2.3.2	Embedding Algorithms	17
2.3.2.1	Least Squares on the Distances Embedding . .	18
2.3.2.2	Metric Scaling Embedding	23
2.3.2.3	Similar Performance between Metric Scaling and Least Squares on the Distances in a US Map Reconstruction	25
2.4	Concluding Remarks	27
3	PCA, LDA and Their Embedding Interpretations	29
3.1	Introduction	29
3.2	PCA	31
3.2.1	Classical PCA	31
3.2.2	PCA for Discriminant Analysis	33
3.3	LDA	34
3.4	PCA and LDA: The Embedding Interpretations	35
3.4.1	The Duality between PCA and Metric MDS	36
3.4.1.1	PCA: Least Squared Fitting in $\mathbb{R}^v$	36
3.4.1.2	The Duality	38

3.4.2	LDA: All Extreme Case of Three-way Metric Scaling . . .	39
3.4.2.1	Implementation of Embedded Version of LDA . . .	41
3.5	Projecting an Instance into The Space	41
3.6	Discussions	43
4	Kernel Generated Metric for Embedding	45
4.1	Introduction to Kernel Learning	45
4.2	Kernel PCA and Its Embedding Interpretation	47
4.2.1	PCA in Feature Space	48
4.2.2	Embedding Interpretation of KPCA	53
4.3	Kernel LDA	55
4.4	Convolution Kernel Generated Metric for Embedding	58
4.4.1	Tensor Product and Direct Sum Kernels	59
4.4.2	Construction of Distance Matrix	59
4.5	Experiments	62
4.5.1	Experimental Settings and Results	62
4.5.2	Observations and Reasoning	64
4.6	Discussions	67
5	Manifold Embedding: ISOMAP and LLE	69
5.1	Introduction	69
5.2	Performing ISOMAP	72
5.3	Performing LLE	74
5.4	Experiments	77

5.4.1	Experiment I: Spiral Data	78
5.4.2	Experiment II Two Objects from COIL-100	79
5.5	Discussions	81
5.5.1	Pros and Cons of ISOMAP and LLE	81
5.5.2	Connection to The Framework of This Thesis	82
5.5.3	Future Directions	83
6	Embedding Techniques for Face Recognition	87
6.1	Review of Face Recognition	87
6.2	Validity of Embedding Techniques for Face Recognition	89
6.3	Experiments	92
6.3.1	The Face Database	92
6.3.2	Experimental Results	93
6.3.3	Observations and Reasoning	95
6.4	Discussions	96
6.4.1	Theoretical Analysis	97
6.4.2	Empirical Analysis	102
7	Conclusions & Recommendations	104
7.1	Conclusions	104
7.2	Recommendations for Future Research	108
	Bibliography	110

List of Figures

1.1	A conventional pattern recognition system as described in [1] and the feature extraction step is also depicted as a dimensionality reduction step	2
1.2	A one-dimensional manifold (a segment of spiral) $\mathbf{M}$ in $\mathbb{R}^3$. . .	4
2.1	A simplified concept of majorization, $f(x)$ is the target function, and $g(x)$ is a surrogate function at a certain stage.	19
2.2	Reconstructed maps from the distances of major US cities . . .	26
3.1	(a) The first principal component contains most information for discrimination. (b) The first principal component contains no discriminant information	33
3.2	Pearson's least square fitting model of PCA	37
4.1	Relationships between the Intra-class and Inter-class distances in Tensor Product and Direct Sum Mapped Feature Space	61
4.2	Classification accuracy on six UCI data sets. In each panel, from left to right, the four bars record the KPCA, Embedded KPCA, KLDA, and convolution kernel embedding, respectively.	65

5.1	Derived one dimensional embedding from a 3D spiral shown in Fig. 1.2 using PCA or metric MDS using Euclidean distances. .	78
5.2	Derived one dimensional embedding from a 3D spiral shown in Fig. 1.2 using ISOMAP and LLE.	79
5.3	Two objects are embedded using ISOMAP and LLE with the neighborhood factor $k = 6$	84
5.4	36 cat toys, and the pose interval is 10 degrees.	85
5.5	36 duck toys, and the pose interval is 10 degrees.	86
5.6	One-dimensional manifolds for both the cat and duck objects are plotted in one diagram.	86
6.1	An ideal example of allocating three persons into a plane (2-dimensional space), where these three persons are separated clearly, and images from the same person are clustered tightly.	91
6.2	Recognition accuracy of nine methods on two databases. In each panel, from left to right, the nine bars record the eigenface, embedded eigenface, fisherface, embedded fisherface, protoface, kernel eigenface, embedded version of kernel eigenface, kernel fisherface, and convolution kernel embedding, respectively.	94
6.3	Two hypothetical distributions denote two classes, one is for the total scatter, and the other is for the between-class scatter. The first three principal components from these two distributions are shown in a single 3-dimensional scatter plot. The circles for between-class scatter and the crosses for total scatter.	102

List of Tables

2.1	The driving distances between 13 US major cities in miles . . .	25
4.1	Example of mercer kernel functions	46
4.2	Average recognition accuracy for six UCI data sets	64
6.1	Average recognition accuracy for AT&T and Yale face databases	94
7.1	Distance metrics used in dimensionality reduction techniques . .	105

Chapter 1

Introduction

1.1 Motivation of Dimensionality Reduction

Dimensionality reduction is a crucial step in some data analysis tasks, where every data sample is represented as a real-valued vector. Take a generic pattern recognition task as an example. Given a set of data samples $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\} \subset \mathcal{D}^v$ with the class label for each $\mathbf{x}_i$, $t_i \in \{1, 2, \dots, g\}$, a rule must be learned from these v -dimensional vectors $\mathbf{x}_i$'s and their corresponding class label t_i 's, such that every new coming vector $\mathbf{x}$ can be assigned with a correct class label $t \in \{1, 2, \dots, g\}$. In such a task, a data sample i is represented by a vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{iv})$, and typically, the dimension v is very high. The reason is that it is hard to select only a portion of the v number of measurements, and a conventional way is taking all the measurements to denote a sample. According to [1], doing so can guarantee that no information will be lost, but this will usually result in high dimensionality of the vector.

Representing data samples as high dimensional vectors presents difficulties

1.1. Motivation of Dimensionality Reduction

to data analysis systems. The first obvious one is that it will challenge the computation capability of a system. Take a typical face recognition problem as an example, we are given a set of greyscale face images with $m \times n$ pixels each. In practice, we could have $m = n = 256$, or each image is represented by a 65536-dimensional vector. If these images are classified using a multilayer perceptron neural network classifier system, the number of weights would be exceedingly large, and it is just intractable for most computers. As a result, the dimensionality of all the images is often reduced before performing any further classification. According to some pattern recognition books [1][2][3], there is a step named Feature Extraction, which is essentially a dimensionality reduction step. Through this step, the high dimensional data samples are represented by lower dimensional vectors, such that the system is able to handle those samples.

Fig. 1.1 diagrammatically gives an illustration of such a pattern recognition system.

Figure 1.1: A conventional pattern recognition system as described in [1], and the feature extraction step is also depicted as a dimensionality reduction step

The second reason is due to the well-known problem of Curse of Dimensionality [4], which states a fact that, in the absence of simplifying assumptions, in order to estimate a function of several variables to a given variance, the required

sample size grows exponentially to the number of variables for a good estimation. In other words, for most data analysis systems, if the dimensionality of the data samples increases, the number of data samples needed in training the system increases exponentially. In practise, from time to time, to obtain a sufficient number of data samples carries a big burden, and sometimes it is even infeasible. In order to train a reliable system with a limited number of samples, it is good to keep the dimensionality of the data samples low. Hence, finding a lower dimensional space that can be used to represent the data well for the intended task is important.

Another objective of dimensionality reduction is based on a phenomenon that some complex and high-dimensional data are actually governed by only a few simple variables, and the data samples are spread with a nonlinear structure in a manifold. Dimensionality reduction techniques are used to "model" the data samples, or "decrypt" the data spread. Fig. 1.2 illustrates a good example, where every data sample is a point in a spiral manifold in a 3-dimensional space, and this manifold is intrinsically only governed by one variable. In other words, all the data samples can essentially be denoted by just one variable instead of three. Hence, finding a mapping that discovers the structure-preserving lower dimensional representation is important.

1.2 A Brief Review of Some Dimensionality Reduction Techniques

Principal Component Analysis (PCA) [5] [6] and Linear Discriminant Analysis (LDA) [2] are two commonly used techniques to accomplish dimensionality

Figure 1.2: A one-dimensional manifold (a segment of spiral) $\mathcal{M}$ in $\mathbb{R}^3$

reduction. However, they are not designed to discover the structure of data spread. Recently, two methods are reported to model the data spread or manifold, and they are ISOMAP [7] and LLE [8].

Both PCA and LDA are classical multivariate analysis techniques, the success in reducing dimensionality by both methods implies that the dimensionality reduction can be achieved because there exist correlations between the variates of any sample vector, i.e., for $\mathbf{x}$, the v elements are not mutually independent, and there are redundancies, so it is possible to reduce the dimensionality from v to p , where $p < v$. To find the redundancies, a certain number, n , samples $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ are required. Though PCA and LDA are both multivariate analysis techniques, they are very different in terms of the objectives they want to achieve besides that both of them reduce the dimensionality. PCA is used to preserve as much as the variance while reduce the dimensionality, LDA is to find a few dimensions which can separate the data samples as clear as it can be, according to the classes.

Dimensionality reduction can also be achieved through another family of techniques, Multidimensional Scaling (MDS) [9] [10], whose purpose is to preserve the known pairwise dissimilarities from a high dimensional space to a lower dimensional space. MDS lies in the belief that a p dimensional space is of enough capacity to accommodate n samples, at the same time the pairwise dissimilarities are preserved. In other words, this family of techniques analyzes the distances between every two samples, and embeds all samples to a lower dimensional space. Also noted that there is a duality between the Metric MDS and PCA [9] [11], from which PCA can be interpreted as to embed all the given samples while preserving all pairwise Euclidean distances. The main theme of this thesis is to introduce a framework derived from MDS, based on which most conventional dimensionality reduction techniques can be interpreted within this framework. In order to serve this main theme, I will show that LDA can also be interpreted under Metric MDS embedding, in which only all the class means are embedded, and the preserved dissimilarities are the Mahalanobis distances. Interpreting both PCA and LDA using metric MDS technique will introduce a natural link between these two classical dimensionality reduction techniques.

Since the 1990s, a family of techniques known as Kernel methods [12] [13] have attracted attention from the machine learning and pattern recognition community. More recently, both PCA and LDA have been generalized in the context of kernel by Scholkopf et al. [14] and Baudat and Anouar [15]. Applying the kernel trick to distance [16], the kernel PCA is able to be interpreted by metric MDS with a particular distance metric [17] determined by the specific kernel function used. Kernel PCA does not utilize the class information just as in the conventional PCA. In order to gain discriminant power from the class information for pattern classification, Zhang et al. [18] [19] proposed a method

where the class information is imposed on the distance metric by means of a convolution kernel between the sample vectors and the class labels. As a result, the distances between intra-class and inter-class data samples are separated by a threshold. Metric MDS is carried out to embed the data samples in a lower dimensional space.

As mentioned in Section 1.1, there is another objective-driven dimensionality reduction method, which is designed to model a manifold. None of PCA, LDA and their kernel versions are suitable for this task, Tenenbaum et al. [7] [20] derived a new technique named ISOMAP, which is essentially an extension of the metric MDS. Although as a mathematical term, a manifold is a complicated concept (for details, readers can refer to a book by Alan [21]), from an engineering understanding, however, it can be simplified into two points [20] [8].

1. Globally, data spread is continuous, and the connection is nonlinear
2. Locally, the data samples spread linearly

Observing these two points, ISOMAP first determines the neighborhood for every data samples. Then, it calculates the Euclidean distances for those two points belonging to the same neighborhood. After that, it applies the Dijkstra's algorithm to compute the shortest distances for those points which are not neighbors. Finally, it embeds all the data samples according to the distances calculated above through metric MDS. In contrast to ISOMAP's explicit application of different distance metrics for neighboring and non-neighboring points, LLE [8] implicitly made the two distance metrics different to reduce the nonlinear manifold.

1.3 Objective and Major Contributions of This Thesis

The goal of this research is to review and investigate some state-of-the-art dimensionality reduction techniques, and to interpret some conventional techniques from an embedding point of view linked by MDS. Links between some conventional techniques and the family of embedding techniques are established. There exist some intrinsic links even between those conventional techniques. An embedding based dimensionality reduction framework is used in this thesis to link all the chapters. The framework selects some certain representatives from a given training set, then works out a squared distance matrix D^2 , finally a metric embedding process is conducted to embed all the representatives into a lower dimensional space. The main contributions of this research are summarized below:

1. A Unified Dimensionality Reduction Framework based on Data Embedding

Based on data embedding using metric MDS, a unified dimensionality reduction framework is developed in this thesis. This framework is the main theme of this thesis. Under this framework, there are two controlling factors. One is how to select representatives from the training samples, and the other is what distance metric will be used in calculating the distance matrix D^2 of the selected representatives. The selection of the representatives and the distance metric may make the derived lower dimensional space either with more generative or more discriminative sense. Normally, in order to gain more discriminative power, the distance metric will involve

the given class information of the training samples and try to separate the classes as widely as possible. In order to be more generative, all the training samples are selected in building the embedding space.

2. Links between PCA and LDA

In literature, PCA and LDA are both commonly used dimensionality reduction techniques. However, in most circumstances, these two techniques are clearly serving different purposes. As such, they are compared rather than linked. Under the framework of this thesis, these two techniques live under one roof except that they select different representative distance metrics in embedding.

3. Embedding interpretations of kernel methods

Kernel methods have achieved a big success in machine learning and pattern recognition. Through the kernel trick of distances, most kernel methods can be interpreted in the context of embedding. In this thesis, I introduce the embedding interpretation of kernel PCA, which has also been reported by Williams [22] recently. Convolution kernels are able to combine two different domains into a single distance metric, which can be generalized for many discriminant analysis tasks. Again, the embedding interpretations can fit into the framework of this thesis.

4. ISOMAP and LLE, different metrics for neighboring and non-neighboring data points

ISOMAP and LLE are reviewed in this thesis with emphasis that both of them reduce dimensionalities by data embedding. In contrast to the PCA and LDA's embedding interpretations, ISOMAP and LLE have different metrics for the local and global points. In other words, in PCA and LDA,

1.4. Organization of This Thesis

the metrics are homogenous, but ISOMAP and LLE have heterogenous metrics. The other finding is that ISOMAP can be easily categorized into the embedding framework proposed in this thesis but it is hard to put LLE under the framework.

5. Face recognition as an application of embedding techniques

Since eigenface (based on PCA) and fisherface (based on LDA) have been widely used as benchmarks in face recognition, face recognition is chosen to be an application to demonstrate all the techniques discussed in this thesis, either in the conventional context or in the embedding context. Under the framework, tunable distance metrics can be easily applied to generalize the capability of a recognition system, but conventional methods may not be easy to be generalized. In addition, it is shown that for both AT&T and Yale databases, the class means are sufficient for training the face space.

1.4 Organization of This Thesis

In this thesis, I reviewed some dimensionality reduction techniques, either to extract features for data analysis or to model the data manifold. Conventionally, those techniques for feature extraction mostly rely on the analysis of variance. I intended to interpret those techniques from an embedding point of view through establishing the links between them and the metric MDS. The techniques modelling the manifold are generally based on embedding techniques.

In Chapter 2, I discuss a Euclidean distance preserving embedding technique, Metric Multidimensional Scaling, which will be the basic embedding technique for the whole thesis.

Following that, Chapter 3 demonstrate that both PCA and LDA can be interpreted from an embedding point of view, with different distance metrics and selections of embedding samples.

In Chapter 4, Kernel generated metrics are discussed in detail for embedding. Kernel PCA proposed by Scholkopf et al. [14] are linked to embedding techniques through the kernel trick for distances [16]. In addition, some more generalized kernel distance metrics based on Zhang et al.'s work [23] are also discussed.

Chapter 5 is devoted to a special class of dimensionality reduction techniques that models the data manifold. This chapter discusses the ISOMAP technique proposed by Tenenbaum et al. [7] and Locally Linear Embedding (LLE) technique proposed by Roweis and Saul [8].

Chapter 6 is on experimental evaluations of these related methods. Nine sets of experiments on two face databases are conducted, and the results are reported and analyzed.

Finally, Chapter 7 concludes the whole thesis with some recommendations for possible future directions.

Chapter 2

Multidimensional Scaling

2.1 Introduction

Given a set of n observations, and the pairwise dissimilarities δ_{ij} between any two observations i and j , Multidimensional Scaling (MDS) constructs a geometric representation of all those observations. The dissimilarities¹, which are distance-like quantities, reveal strong relationships, either qualitatively or quantitatively. The final representation Y , a $n \times p$ matrix, coordinates all the observations in space $\mathbb{R}^p$, normally Euclidean. In $\mathbb{R}^p$, one point represents one observation, and the distances between the points, d_{ij} , will match, as closely as possible, the original dissimilarities δ_{ij} .

In literature, MDS was invented and reinvented, developed and redeveloped in many different disciplines [10]. Using the distances between a set of cities to reconstruct a map is one of the classic ways to illustrate the MDS. In 1982, Greenacre and Underhill [24] used flying times between South African airports,

¹Sometimes, the information can also be the similarities, which are inversely related to distances.

and in 1979, Mardia et al. [25] used the road distances between some British cities to successfully construct the South African and British maps respectively. MDS has also been heavily investigated in some other research disciplines such as the Economic Geography, Genetics and Systematic Zoology [26], Psychology and Psychometric [27], Political Science, and the Biochemistry [28] [29]. Furthermore, there is a set of very detailed books on MDS, such as those by Cox and Cox's [9] and Borg and Groenen's [10].

After a brief literature review of the MDS, the rest of this chapter is organized as follows. In Section 2.2, MDS is categorized into different types. Then, some widely used reconstruction loss functions and embedding algorithms for Metric MDS are reviewed in Section 2.3. In the same section, a map reconstruction experiment is used to show the similar performances between two embedding algorithms, namely metric scaling embedding and least squares on the distances embedding.

2.2 Types of MDS

There are two different forms of MDS, depending on how strong the given dissimilarities are. In some cases, the dissimilarities are known numbers, to some extent, can be treated as the distances. In other cases, only the rank order is given.

2.2.1 Metric Scaling

In metric scaling the dissimilarities between all observations are known numbers, and they can be approximated by distances. The observations are mapped into

a metric space, where the distances match the dissimilarities as faithfully as possible. From a mathematical viewpoint, metric scaling is done in two steps. Firstly, some certain loss functions are defined. Then, an optimization process is performed to find the "best-fit" space in which the distances are preserved as much as possible.

Noted that in metric scaling, the distance in the new metric space may be in a functional form with the corresponding dissimilarity.

$$d_{ij} = f(\delta_{ij}) \quad (2.1)$$

where f is a specified (analytic) function². For example, the negative exponential function was used by Shepard [30] in his early theory of recognition, and the polynomial transformations were suggested by Kruskal and Wish [31].

In this thesis, one important objective is doing dimensionality reduction by using MDS. Just as mentioned in Chapter 1, the original v -dimensional data is known, in other words, the original pairwise dissimilarities are known. To find a lower p -dimensional space, metric scaling can be a straightforward option.

2.2.2 Nonmetric Scaling

From time to time, in certain disciplines such as psychology, the relationships between observations can only be nominal or ordinal. Nominal data are classificatory, which means that only different classes are distinguishable, for example, the color of eyes. Ordinal data are qualitative rather than quantitative, for example, a tiger is bigger than a home cat.

²Actually in the following loss functions, f is chosen to be an identical function, as a result, every δ is just approximated by the corresponding d .

The other case of nonmetric scaling is due to incomplete information. There are only partial dissimilarities collected for some reasons such as the natural restrictions for observing distances between certain observations. During the scaling, the new metric space is constructed from the partial information; meanwhile the missing distances are estimated.

2.2.3 Three-way Scaling

Three-way scaling deals with information on dissimilarities between n observations for m subjects. Or there are m $n \times n$ dissimilarities matrix available, and the objective is to find a space to accommodate all the n observations while considering the m subjects.

For this problem, there are two extreme solutions. One is to perform a separate MDS for each subject, and the other is to perform one single MDS for the average occasion. Following these two extreme cases, three-way scaling is concerned on building a strategy between the two cases.

Three-way scaling computes m MDS solutions, while they are required to be related to each other. For example, with a tunable transformation function f (cf Eq. 2.1) for each configuration, every MDS space can be identical. A very detailed introduction to this technique can be found in [32].

2.3 Embedding Algorithms for Metric MDS

For the main theme of this thesis, I will mainly introduce the metric scaling techniques here. In fact, nonmetric scaling algorithms are mostly motivated from the metric scaling counterparts. For instance, a typical nonmetric scaling

technique proposed by Kruskal [33] was modified from the least square metric scaling algorithm, which will be discussed in following sections.

2.3.1 Loss Functions

Basically, there are three loss functions applied frequently, and they are the *Least Squares on the Distances* or STRESS, the *Least Squares on the Squared Distances* or SSTRESS, and the *Least Squares on the Inner Products* or STRAIN.

2.3.1.1 STRESS

The most straightforward measurement for fitness is, of course, the least squares loss. In MDS, it is termed as STRESS, and defined by

$$STRESS(Y) \triangleq \sum_{i=1}^n \sum_{j=1}^n w_{ij} (\delta_{ij} - d_{ij}(Y))^2 \quad (2.2)$$

where w_{ij} is the data weights and all sum to 1, which measures the level of importance for fitness of each dissimilarity, and Y is the final configuration, i.e., a $n \times p$ matrix with each row being $1 \times p$ vector.

Obviously, Eq. 2.2 applies to metric scaling only. To deal with the nonmetric data, Kruskal [33] proposed a modified version as

$$STRESS(Y, \hat{D}) \triangleq \frac{\sum_{i=1}^n \sum_{j=1}^n w_{ij} (\hat{d}_{ij} - d_{ij}(Y))^2}{\sum_{i=1}^n \sum_{j=1}^n w_{ij} (d_{ij} - \bar{d}_{ij}(Y))^2} \quad (2.3)$$

To solve this equation and obtain the best fit representation, Kruskal proposed an iterative technique, which is discussed in detail in Cox and Cox's book [9] pp. 69-71. In this equation, the denominator is a scaling factor, and inside

this factor $\bar{d}_{ij}(Y)$ is the average distance obtained from representations at each stage of iteration. $\hat{D}$ is called the *disparity matrix* filled by all *disparities*. In nonmetric MDS, the given dissimilarities are in rank-order, in order to do a distance embedding, some quantitative values are assigned to denote the dissimilarities among those objects. Meanwhile the rank-order shall be preserved. For example, if we are given such information that A is taller than B, and B is taller than C. Then we know that the quantitative dissimilarity between A and C will be greater than the dissimilarity between A and B. Those dissimilarity values to replace the rank-order information are called *disparities*. This STRESS function can be minimized over both Y and $\hat{D}$ using the iterative technique mentioned above. At the final stage of the iteration, both a best disparity matrix $\hat{D}$ and a good embedding estimation Y are obtained. And the process to obtain a set of quantitative values to denote the dissimilarities while keeping the rank-order unchanged is called the isotonic regression.

Noted that STRESS is not differentiable if there are two points coincide because $d_{ij}(Y)$ is a square root function, which is not differentiable at 0.

2.3.1.2 SSTRESS

The other popularly used loss function is the least squares on the squared distances, defined as

$$SSTRESS(Y) \triangleq \sum_{i=1}^n \sum_{j=1}^n w_{ij} (\delta_{ij}^2 - d_{ij}^2(Y))^2 \quad (2.4)$$

where the squared distances are taken as a measure of fitness. It is clear that this loss function is a fourth-order polynomial, so the function should be smooth. However, there are often a large number of local optimal points.

Same as Eq. 2.2, this SSTRESS is for metric scaling only. To extend this form to fit a nonmetric scaling problem is quite straightforward by using the same technique as in Eq. 2.3.

2.3.1.3 STRAIN

In spite of the least squares on measurements of distances (including squared distances), another very popularly used loss function is based on the minimization of the inner products, and defined as

$$STRAIN(Y) \triangleq tr\{H(\Delta^2 - D^2(Y))H(\Delta^2 - D^2(Y))H\} \quad (2.5)$$

where H is the centering matrix, and defined as $H = I - \frac{1_n 1_n'}{n}$, inside it, I is a properly sized identity matrix, and 1_n is a n -sized column vector filled with all 1's. $\Delta^{(2)}$ is the squared dissimilarities matrix, $D^{(2)}(Y)$ is the expected squared distances matrix in the reconstructed space, Y . This equation is minimized over Y .

The functional difference as compared to the previous two loss functions is described in Section 2.3.2.2, where metric scaling embedding is discussed in detail to show the process of minimization of the STRAIN function.

2.3.2 Embedding Algorithms

With the loss functions known, to reconstruct a space becomes an optimization problem, or the mere purpose now is to minimize the loss functions. In this section, I will discuss the embedding algorithms based on the STRESS and STRAIN loss functions in details, which corresponds to the *least squares* on the

distances embedding and the metric scaling embedding, respectively.

2.3.2.1 Least Squares on the Distances Embedding

The loss function for the least squares on the distance embedding is of course the STRESS as defined in Eq. 2.2. Kruskal proposed the very first algorithm in 1964 in [33], where the gradient methods with *elaborate* (or not constant) step-size procedures were applied to minimize the STRESS. Due to the complexities of the gradient based algorithms and the possible singularity in STRESS as mentioned in Section 2.3.1.1, de Leeuw [34] introduced the *majorization method*.

Majorization method recently has attracted enormous interests, and appeared under a set of names, *iterative majorization*, *optimization transfer*, *block-relaxation algorithm*, *Surrogate Maximization*, and *Majorize-Minimize or Minorize-maximize* [35] [36] [37] [38] [39]. The feature of this set of algorithms is that assuming you need to minimize the target function, it finds a surrogate function at each stage, and this surrogate function will always be greater than or equal to your target function. Then if you find the minimum value of the surrogate function, this value will of course be the upper bound of your target function at this stage. As shown in Fig. 2.1, at a certain iterative stage, function $g(x)$ is chosen as the surrogate function for the target function $f(x)$.

Because normally the target function is not differentiated directly, it needs not to be differentiable everywhere, just as the function $f(x)$ shown in the Fig. 2.1, needs not to be differentiated everywhere. If the surrogate function is properly selected, it can be differentiated and extreme points can be easily found, and most likely, we will find such surrogate functions. The function $g(x)$ in the Fig. 2.1 is a good example. Furthermore, the convergency can be proved

Figure 2.1: A simplified concept of majorization, $f(x)$ is the target function, and $g(x)$ is a surrogate function at a certain stage.

using convex analysis methods.

The standard majorization algorithm on the Eq. 2.2 or the STRESS function can be found in many textbooks, such as in [10]. Recall our problem statement on dimensionality reduction, we are given an $n \times v$ data matrix $\mathbf{X}$, which denotes n observations in a higher v -dimensional space, and we are asked to find an $n \times p$ data matrix $\mathbf{Y}$ with $p < n$. Assume there is a projection matrix $\mathbf{P}$ with size of $v \times p$, then $\mathbf{Y} = \mathbf{XP}$, so the task is to find the projection matrix $\mathbf{P}$.

Targeting on the projection matrix $\mathbf{P}$, a modified majorization algorithm (similar to Zhang's iterative algorithm for distance metric [23] but with different function $B(\mathbf{P})$ as in the iterative update equation, Eq. 2.14) is introduced here. Assume the data weights w_{ij} in Eq. 2.2 is the same with a unity value, or we take all the dissimilarities measurements equally, at the same time, also assume

In this thesis, all the configuration matrix is formed to be a collection of a number of row vectors, as such, the projection matrix is the right multiply matrix.

the given dissimilarity matrix is symmetric⁴, Eq. 2.2 is modified to

$$S = \sum_{j=1}^n \sum_{i < j} (\delta_{ij} - d_{ij}(Y))^2 \quad (2.6)$$

Since the $\sum_{j=1}^n$ is always in the front, it can be simply dropped to make the equation cleaner, and expanding the RHS,

$$S = \sum_{i < j} \delta_{ij}^2 + \sum_{i < j} d_{ij}^2(XP) - 2 \sum_{i < j} \delta_{ij} d_{ij}(XP)$$

Because the given $\mathbf{X}$ is a constant, minimizing the above function is equivalent to minimizing

$$S = \eta_\delta^2 + \eta^2(P) - 2\rho(P) \quad (2.7)$$

Since the first part of the above equation purely depends on the fixed dissimilarities δ_{ij} , it is a constant. The second part, $\eta^2(P)$, is the sum of the squared distances $d_{ij}^2(Y) = d_{ij}^2(XP)$. The final part, $\rho(P)$, is the sum of weighted distances $\delta_{ij} d_{ij}(XP)$. To obtain the $\mathbf{P}$, let us consider the $\eta^2(P)$ and $\rho(P)$ separately.

Firstly,

In this thesis, the dissimilarities are obtained from a given data set $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ using different distance metrics, such as Euclidean distance and Kernel generated distance, as a result, the dissimilarity matrix is symmetric.

$$\begin{aligned}
\eta^2(P) &= \sum_{i < j} d_{ij}^2(XP) \\
&= \sum_{i < j} (x_i P - x_j P)(x_i P - x_j P)' \\
&= \sum_{i < j} (x_i - x_j) P P' (x_i - x_j)' \\
&= \text{tr}(P' V P)
\end{aligned} \tag{2.8}$$

where $V = nX'HX$, here H is again the centering matrix.

To prove the correctness of this equation, a compact expression for the sum of squared distances shall be briefly explained first, and the detailed discussion of this concept can be found in [9] PP. 151. Given a representation Y of size $n \times p$, representing n number of $1 \times p$ row vectors, the squared distance $d_{ij}^2(Y)$ between two vector y_i and y_j is

$$d_{ij}^2(Y) = \text{tr}(Y' A_{ij} Y)$$

The matrix A_{ij} is simply a matrix with $a_{ii} = a_{jj} = 1$, $a_{ij} = a_{ji} = -1$, and all other elements zero. So the summation of all the squared distances becomes $\text{tr}\{Y'(\sum_{j=1}^n \sum_{i < j} A_{ij})Y\}$. Noticed that $\sum_{j=1}^n \sum_{i < j} A_{ij} = nH$,

$$\sum_{i < j} d_{ij}^2(XP) = \text{tr}(XP)' nH (XP) = \text{tr} P' (nX' H X) P \tag{2.9}$$

It is clear that the Eq. 2.8 is correct.

Next,

$$\begin{aligned}
\rho(P) &= \sum_{i < j} \delta_{ij} d_{ij}(XP) \\
&= \sum_{i < j} \frac{\delta_{ij}}{d_{ij}} d_{ij}^2(XP) \\
&= \sum_{i < j} \frac{\delta_{ij}}{d_{ij}} (x_i - x_j) P P' (x_i - x_j)' \\
&= \text{tr}(P' B(P) P)
\end{aligned} \tag{2.10}$$

where $B(P) = n \frac{\delta_{ij}}{d_{ij}} X' H X$.

then,

$$S(P) = \eta_\delta^2 + \text{tr}(P' V P) - 2 \text{tr}(P' B(P) P) \tag{2.11}$$

A majorization function, T , for loss function S is defined as

$$T(P, U) = \eta_\delta^2 + \text{tr}(P' V P) - 2 \text{tr}(P' B(U) U) \tag{2.12}$$

To minimize T , differentiate T with respect to the U first, and then set it to zero.

$$\frac{\partial T}{\partial U} = 2VP - 2B(U)U = 0 \tag{2.13}$$

So, P can be iteratively updated by

$$P_{\text{new}} = V^+ B(P_{\text{old}}) P_{\text{old}} \tag{2.14}$$

where V^+ is the Moore-Penrose inverse [40] of V .

2.3.2.2 Metric Scaling Embedding

In the above, an iterative algorithm is introduced to find the mapping from a higher dimensional space to a lower dimensional space by means of least squares on distances scaling. However, when STRAIN defined in Eq. 2.5 is taken as the loss function, classical metric scaling is preferred due to its analytical form and non-iterative nature.

Squared distance between any two instances y_i and y_j is given by

$$d_{i,j}^2 = \|y_i - y_j\|^2 = y_i' y_i + y_j' y_j - 2y_i' y_j \quad (2.15)$$

In matrix form, it can be written as

$$D^2(Y) = \mathbf{c}\mathbf{1}' + \mathbf{1}\mathbf{c}' - 2YY' \quad (2.16)$$

where $\mathbf{c}$ is a column vector containing the diagonal elements of YY' , and $\mathbf{1}$ is a column vector filled with all 1's. Double centering with the centering matrix,

$$\begin{aligned} HD^2(Y)H &= H(\mathbf{c}\mathbf{1}' + \mathbf{1}\mathbf{c}' - 2YY')H \\ &= H\mathbf{c}\mathbf{0}' + \mathbf{0}\mathbf{c}'H - 2HYY'H \\ &= -2YY' \end{aligned} \quad (2.17)$$

The Eq. 2.17 clearly shows that $HD^2(Y)H = -2YY'$ for any set of data Y , also noted that the centering matrix has such a property that $H = H' = HH' = HH$, the STRAIN function as shown in Eq. 2.5 can be rewritten as

$$\begin{aligned}
\text{STRAIN}(Y) &\triangleq \text{tr}\{H(\Delta^2 - D^2(Y))H(\Delta^2 - D^2(Y))H\} \\
&= \text{tr}\{(H\Delta^2H - HD^2(Y)H)(H\Delta^2H - HD^2(Y)H)\} \\
&= \text{tr}\{(H\Delta^2H - HD^2(Y)H)'(H\Delta^2H - HD^2(Y)H)\} \\
&= 4\text{tr}\{(YY' - XX')'(YY' - XX')\} \tag{2.18}
\end{aligned}$$

In matrix calculation, given a function $f(B) = \text{tr}(B - A)'(B - A)$, if we want to minimize this function over B with known A , we shall first work out the gradient $\nabla f(B)$ and then set $\nabla f(B) = 0$. Expanding the function $f(B)$, we obtain $f(B) = \text{tr}B'B + \text{tr}A'A - 2\text{tr}B'A$, so the gradient $\nabla f(B) = \nabla \text{tr}B'B + \nabla A'A - 2\nabla \text{tr}B'A = 2B + 0 - 2A = 2B - 2A$. Set the gradient to 0, it is clear that when $B = A$, the function $f(B)$ hits the minimum. Above Eq. 2.18 has a similar form with function $f(B)$, so to minimize the STRAIN, we need to set $YY' = XX'$. Also noted that $XX' = -\frac{1}{2}H\Delta^2H$, so the $B = YY'$ can be approximated as $B = -\frac{1}{2}H\Delta^2H$. If B is positive semi-definite matrix with rank r , B can be eigen-decomposed into

$$B = U\Lambda U' \tag{2.19}$$

where $U = [u_1, u_2, \dots, u_r, \dots, u_m]$, a matrix with a set of eigenvectors u listed column by column, and the diagonal elements of Λ is filled by the eigenvalues.

So the basic coordinates of the reconstructed space will be

$$Y = U\Lambda^{1/2} \tag{2.20}$$

As the coordinates are generally non-negative number, the B shall be positive

Table 2.1: The driving distances between 13 US major cities in miles

	SE	SF	LA	LV	WI	DA	HO	LO	CH	AT	NY	WA	MI
SE	0	810	1130	1180	1860	2130	2370	2140	2050	2630	2840	2720	3300
SF	810	0	390	570	1700	1750	1910	2120	2170	2480	2930	2840	3090
LA	1130	390	0	270	1390	1400	1540	1840	2050	2190	2790	2650	2720
LV	1180	570	270	0	1180	1230	1470	1620	1780	1980	2570	2420	2570
WI	1860	1700	1390	1180	0	370	620	460	730	950	1420	1280	1600
DA	2130	1750	1400	1230	370	0	250	660	920	820	1560	1310	1340
HO	2370	1910	1540	1470	620	250	0	780	1090	790	1610	1370	1190
LO	2140	2120	1840	1620	460	660	780	0	290	570	980	860	1230
CH	2050	2170	2050	1780	730	920	1090	290	0	710	810	710	1400
AT	2630	2480	2190	1980	950	820	790	570	710	0	850	620	660
NY	2840	2930	2790	2570	1420	1560	1610	980	810	850	0	240	1330
WA	2720	2840	2650	2420	1280	1310	1370	860	710	620	240	0	1060
MI	3300	3090	2720	2570	1600	1340	1190	1230	1400	660	1330	1060	0

semi-definite. In other words, the dissimilarities matrix Δ shall be more metric like.

2.3.2.3 Similar Performance between Metric Scaling and Least Squares on the Distances in a US Map Reconstruction

In this section, I will reconstruct two maps based on a set of driving distances between 13 US major cities using the least squares on distance embedding and metric scaling embedding, respectively. Empirically, I would like to show the similar performance between these two embedding methods, namely metric scaling embedding and least squares on the distances embedding. The distances are obtained from the following web site:

<http://www.travelnotes.org/NorthAmerica/distances.htm>

In Table 2.1, approximated driving distances between 13 US cities are recorded in terms of miles. The notations are as follows. SE: Seattle, SF: San Francisco,

LA: Los Angeles, LV: Lw Vegas, WI: Wichita, DA: Dallas, HO: Houston, LO: St. Louis, CH: Chicago, AT: Atlanta, NY: New York City, WA: Washington DC, and MI: Miami.

Both the least squares on distances scaling and the metric scaling are applied on this set of data, and the resulting maps are shown in Figure 2.2.

(a) The map from classic scaling (b) The map from iterative majorization

Figure 2.2: Reconstructed maps from the distances of major US cities

It is shown that the two maps are almost identical, which is expected. Most importantly, they provide correct reflections on the true related locations for those cities, although both of them actually have been rotated 180 degrees clockwise if compared to the conventional map-making rules, where up means north and left means west.

Theoretically, the two mappings should be similar but with minor differences. Least squares on the distance embedding is based on Eq. 2.2 with the assumption that $w_{ij} = 1$, or to minimize the function $\sum_{i=1}^n \sum_{j=1}^n (\delta_{ij} - d_{ij})^2 = 0$. On the other hand, metric scaling is based on the STRAIN loss function, and minimizes the function $tr\{(YY' - XX')'(YY' - XX')\} = 0$ as shown in Eq. 2.18. Let $B = YY'$ and $A = XX'$, this function can be expressed as

$$\begin{aligned}
0 &= \text{tr}\{(YY' - XX')'(YY' - XX')\} \\
&= \text{tr}(B - A)'(B - A) \\
&= \sum_{i=1}^n \sum_{j=1}^n (a_{ij} - b_{ij})^2
\end{aligned} \tag{2.21}$$

It is very clear that the two methods are minimized based on different criteria, although both methods are minimized over second order polynomials.

2.4 Concluding Remarks

In this thesis, MDS will serve as the fundamental of an embedding based dimensionality reduction framework. There are three steps involved in this framework. First, some meaningful representatives are selected. Second, pairwise squared distances are worked out to form the distance matrix D^2 . Lastly, the metric embedding is conducted on the D^2 to embed all the representatives into a lower dimensional space. Some conventional techniques can be interpreted using this framework. For example, in the next chapter, I will show that PCA selects all the training samples and embeds squared Euclidean distances. On the other hand, LDA only selects class means as representatives, and embeds the squared Mahalanobis distances.

In order to serve the main theme of this thesis, multidimensional scaling is reviewed and investigated at the beginning portion of this thesis. Three commonly used loss functions, STRESS, SSTRESS and STRAIN, and two embedding (scaling) algorithms, least squares on distances and metric scaling are discussed. STRESS is a loss function based on the least squares on distances,

and SSTRESS is based on the least squares on the squared distances. Both STRESS and SSTRESS can be optimized through an iterative process, but in this thesis least squares on distance embedding is only applied on the STRESS loss function. STRAIN, on the other hand, is a loss function based on the dot products, and the optimized configuration can be obtained through the metric scaling. The (dis)similarities of the two scaling methods, or rather the two loss functions, STRESS and STRAIN, are discussed. Finally, a US map reconstruction experiment is conducted to demonstrate the similarities and differences.

Chapter 3

PCA, LDA and Their Embedding Interpretations

3.1 Introduction

One common approach in describing an object is through a long real-valued vector. The underlying rationale of using very long vectors may due to two factors. First, sometimes, it is really difficult to describe an object by just a small number of features. Second, for different tasks, the same feature may have different impacts, and when the task is unknown, a better way is to keep all possible information in the descriptions.

Although a long vector may comprise more information, sometimes it is intractable to compute due to the *curse of the dimensionality* [4]. Commonly, dimensionality reduction is applied to shorten the length of the description. Geometrically speaking, this step embeds the objects from a high dimensional space to a lower dimensional space. Take the information storage (or transmis-

sion) and pattern recognition systems as examples. In a storage or transmission system, the length of a description shall be kept compact so as to save the storage space or transmission bandwidth. The dimensionality reduction process is referred to as data compression or coding. On the other hand, for a pattern recognition system, we need to reduce the dimensionality of the feature vector because long descriptions need complex classifiers, which may worsen the performance [41] [1]. The process is known as the feature extraction or selection.

As mentioned above, most likely, the approaches of dimensionality reduction are constrained by the tasks on hand. There are basically two guidelines and approaches for different purposes. The first one is for purposes like the information storage and transmission systems, where the reduced descriptions are required to be reconstructed back to the original descriptions as faithfully as possible, because finally, we need the original information when we retrieve it either from a hard-disk or a transmission channel. The second one is for purposes like the pattern recognition systems, where classification rates need to be as high as possible, i.e. the objects from different classes need to be separated as clearly as possible in the reduced dimensions.

With respect to the two purposes stated above, the following two sections will be devoted to the principal component analysis (PCA) and linear discriminant analysis (LDA). PCA reduces dimensionality while minimizing the reconstruction error, meanwhile LDA projects data on the most discriminant directions. The title of this chapter expects the discussions about the relations between some dimensionality reduction techniques and embedding techniques. In Section 3.4, PCA and LDA will be expressed in the context of MDS, which is basically an embedding technique. More importantly, I believe that this is the

first attempt to interpret LDA in terms of MDS. And by doing so, PCA and LDA can be categorized under a same embedding framework.

3.2 PCA

Principal component analysis [6] [42] is an old statistical technique (Pearson 1901 [5], Hotelling 1933 [43]), and also probably the most widely used dimensionality reduction technique. The popularity may be due to its conceptual simplicity, its analytical property, and the fact that relatively efficient algorithms exist for its computation [44]. In signal processing it is also known as the *Karhunen-Loève* transform.

3.2.1 Classical PCA

Given a cloud of data points in a higher dimensional space, classical PCA tries to find a reduced dimensional space where all the original points can be accommodated subject to the constraint of preserving the variance as much as possible. Essentially, classical PCA is a one-group method.

Suppose we are given a set of data $\mathbf{X}$ of size $n \times v$, PCA attempts to find a few directions, known as principal components, that contain most of the variations of $\mathbf{X}$ if $\mathbf{X}$ is projected onto a space spanned by these principal components. The first principal component is the direction, in which the projection of $\mathbf{X}$ captures the largest amount of variations. The projection on the second principal component captures the second largest amount of variations, and so on. In addition, there is also one more constraint that all the components need to be mutually orthogonal. For some data sets, just a few components may capture

most of the variations, which provides a possible way to describe the data with fewer components, or reduce the dimensionalities. Noted that the components are mutually orthogonal, PCA is therefore also a decorrelation technique.

Since for any column vector w , the variance for X projected to w is $var(Xw) = w^T S_T w$, where S_T is the total scatter matrix, the first principal component is to maximize $var(Xw)$, subject to a normalizing constraint $w^T w = 1$. Define $\Omega = w^T S_T w - \lambda(w^T w - 1)$, where λ is a Lagrange multiplier, and maximize Ω with respect to both w and λ .

Differentiating with respect to w gives $\frac{\partial \Omega}{\partial w} = 2S_T w - 2\lambda w$, and setting it equal to zero gives

$$S_T w - \lambda w = 0 \quad (3.1)$$

So, w is an eigenvector of S_T corresponding to the eigenvalue λ . Premultiply Eq. (3.1) by w^T ; we have $var(Xw) = w^T S_T w = \lambda w^T w = \lambda$, so to maximize $var(Xw)$, we must take the λ with the largest eigenvalue and the corresponding eigenvector w . Add one more constraint that the second principal component is orthogonal to the first principal component, we can easily derive that the second principal component is the eigenvector corresponding to the second largest eigenvalue in (3.1), and so on.

Line up the eigenvectors according to the value of corresponding eigenvalues, in most cases, only a smaller number of eigenvalues are of interest. A way to measure the sufficiency can be set by the ratio between the eigenvalue to the largest eigenvalue, for example, the threshold can be set as 1/1000. Selecting a subset of the principal components, $[w_1, w_2, \dots, w_p]$, to form a projection matrix P , where $p < v$. i.e. the original v -dimensional data X has been embedded into

Figure 3.1: (a) The first principal component contains most information for discrimination. (b) The first principal component contains no discriminant information

a p -dimensional space to form Y , a $n \times p$ matrix.

$$Y = XP \quad (3.2)$$

3.2.2 PCA for Discriminant Analysis

Just as mentioned at the beginning of this chapter, high dimensional data generally need more complex classifiers, i.e., the performance may be worsened if the dimensionality of input is too high. PCA, as a dimension reduction technique, may do favor for preprocessing the data for further discriminant analysis. Both the books by Jolliffe [6] and Jackson [42] discussed this theme in depth. However, just a simple example as in Figure 3.1 may question the suitability of PCA for discriminant analysis.

In Figure 3.1, two 2-dimensional hypothetical distributions are illustrated by two ellipses in two cases. In case (a), the first principal component contains all the discriminant information. In case (b), however, the first principal component contains no discriminant information. Just as what has been argued by Flury

[45] that the success of discriminant analysis using PCA depends on serendipity. Flury, however, also states that, in most cases, PCA is suitable for discriminant analysis on biometric data. In other words, most likely, the biometric data fall into the case (a) in Figure 3.1 according to [45].

One very successful application of PCA in discriminant analysis may be due to the *eigenface* proposed by Turk and Penelaud [46]. Eigenface applied PCA on face images for recognition purposes. It also motivated a family of face recognition approaches, subspace based approaches, which have, to some extent, dominated the face recognition research since the 1990s.

3.3 LDA

PCA finds a subspace to demonstrate the data structure as a single group. On the contrary, LDA targets building a subspace to show the group structure of a set of data. In other words, LDA finds a subspace in which the original data are projected and located separately according to the class labels.

Given a data set X and known group labels, the LDA tries to maximize the ratio of the between-class variance and the within-class variance,

$$\frac{\text{var}(X_B \omega)}{\text{var}(X_1 \omega) + \text{var}(X_2 \omega) + \dots + \text{var}(X_g \omega)} = \frac{\omega^T S_B \omega}{\omega^T S_W \omega},$$
 where g is the number of class, X_B is a $g \times v$ matrix filled by all class means, S_B is the between-class scatter matrix, and the S_W is the within-class scatter matrix. This ratio here is also known as the *fisher criterion*.

One way to solve this maximization problem is to impose some constraints, then utilize the Lagrange multiplier method as used in deriving PCA solutions above. One convenient approach to maximize a fraction is of course to make the

denominator unity, and maximize the numerator. As such, the maximization problem is reduced to maximize $\frac{\omega^T S_B \omega}{\omega^T S_W \omega}$, subject to a constraint that $\omega^T S_W \omega = 1$. Define $\Omega = \omega^T S_B \omega - \lambda(\omega^T S_W \omega - 1)$, where λ again is a Lagrange multiplier. Simply differentiate this equation with respect to w , and set the differential to zero, an eigenequation is given as

$$S_W^{-1} S_B \omega = \lambda \omega \quad (3.3)$$

The same operations are conducted as before in the PCA. We will have a set of eigenvectors corresponding to a set of descending eigenvalues, and these eigenvectors form the components, or bases, of the reduced dimensional space, also known as the fisher space.

3.4 PCA and LDA: The Embedding Interpretations

PCA and LDA are statistical analysis techniques on the variance. On the other hand, embedding techniques are based on the pairwise relationships. Metric MDS is based on one of the strongest relationships, distance. In this section, I will interpret both PCA and LDA as a special Metric MDS. In order to make the interpretation possible, we need to interpret the variance in distance first. A equation given by Gower and Krzanowski [47] plays a key role.

$$\sum_{i=1}^n (\mathbf{x}_i - \bar{\mathbf{x}})^2 = \frac{1}{n} \sum_{i < j}^n (\mathbf{x}_i - \mathbf{x}_j)^2 \quad (3.4)$$

The left-hand side is a measure of the variance for a set of n data points, and the right-hand side is the sum of $\binom{2}{n}$ pairwise distances.

In this section, PCA and LDA will be interpreted from a geometric viewpoint, which is the primitive standing point of all MDS techniques. Before going for details, I would like to give two crude geometric interpretations for a given data matrix X of $n \times v$.

The n observations in $\mathbb{R}^v$. Two observations that are close in space $\mathbb{R}^v$ must have similar values on the v variates. For example, two face images that are close to each other have similar patterns of pixel values.

The v variates in $\mathbb{R}^n$. Two variates that are close in space $\mathbb{R}^n$ implies that they have similar values over the sample of observations. In other words, they are measuring possibly the same object.

3.4.1 The Duality between PCA and Metric MDS

In Section 3.2, the principal components are obtained by maximizing the variance criterion. During the course, the normalizing constraint is also imposed by means of the *Lagrange multiplier*. Here, PCA will be interpreted from an embedding point of view.

3.4.1.1 PCA: Least Squared Fitting in $\mathbb{R}^v$

Each of the n rows of matrix X can be considered a point in $\mathbb{R}^v$. The target is to find a p -dimensional subspace of $\mathbb{R}^v$ such that p is considerably smaller than v and the n points are approximated in this subspace.

We begin with finding the principal one-dimensional space, which is a straight line passing through the origin¹ that provides the best possible fit for all the n points.

Take a look at Fig. 3.2, here, u denotes a unit vector along the first principal dimension, and OH_i is the projection of the vector OM_i on the one-dimensional space. Among the criteria used for fitting a set of n points to a subspace, PCA applied the least squares criterion according to Pearson [5], i.e. the sum of the squared distances $\sum_{i=1}^n M_i H_i^2$ is supposed to be maximized.

Figure 3.2: Pearson's least square fitting model of PCA

Applying the Pythagorean theorem to each of the n right triangles $H_i OM_i$,

$$\sum_{i=1}^n M_i H_i^2 = \sum_{i=1}^n OM_i^2 - \sum_{i=1}^n OH_i^2 \quad (3.5)$$

When all the n points are given, the $\sum_{i=1}^n OM_i^2$ is fixed, so to maximize $\sum_{i=1}^n M_i H_i^2$ is equivalent to minimize $\sum_{i=1}^n OH_i^2$. This value can be expressed

We assume that the data has been centered, or some preprocesses have been applied to make sure that the center of mass will be at the origin.

as a function of $\mathbf{X}$ and $\mathbf{u}$,

$$\sum_{i=1}^n OH_i^2 = (\mathbf{X}\mathbf{u})' \mathbf{X}\mathbf{u} = \mathbf{u}' \mathbf{X}' \mathbf{X} \mathbf{u} \quad (3.6)$$

Note that $\mathbf{u}$ is a unit vector, which means that $\mathbf{u}'\mathbf{u} = 1$. To solve this optimization problem, again applying the Lagrange multiplier, we are actually solving the following equation

$$\mathbf{X}' \mathbf{X} \mathbf{u} = \lambda \mathbf{u} \quad (3.7)$$

and the eigenvectors corresponding to the largest eigenvalue is the first principal component, the eigenvectors corresponding to the second largest eigenvalue is the second principal component, and so on.

3.4.1.2 The Duality

Suppose we are given a data set $\mathbf{X}$ of size $n \times v$, the total (co)variance S_T is defined as $\mathbf{X}^T \mathbf{X}$, if the data set has been mean corrected. PCA just eigendecomposes the S_T as shown in Eq 3.3 to find its principal axes. Substitute the S_T with $\mathbf{X}^T \mathbf{X}$ in Eq 3.3, we have

$$\mathbf{X}^T \mathbf{X} \boldsymbol{\omega} - \lambda \boldsymbol{\omega} = 0 \quad (3.8)$$

On the other hand, if the Euclidean distance is used in the Metric MDS, with the double centering operation indicated in Section 2.3.2.2, finally Metric MDS eigendecomposes a matrix $\mathbf{B}$, and $\mathbf{B} = \mathbf{X} \mathbf{X}^T$.

$$XX^T \mathbf{u} - \gamma \mathbf{u} = 0 \quad (3.9)$$

If we premultiply a X^T in Eq 3.9, it becomes

$$X^T X (X^T \mathbf{u}) - \gamma (X^T \mathbf{u}) = 0 \quad (3.10)$$

Compare Eq 3.10 to Eq 3.8, we can find that the eigenvalues for these two methods are the same $\lambda = \gamma$, while the eigenvectors are related by a constant factor X^T (if data are given, X^T will be a constant).

As a conclusion, if the dissimilarity in a Metric MDS is defined as Euclidean distance, the Metric MDS is exactly the same as a PCA. This phenomenon is known as the duality of PCA and Metric MDS [9] [11].

3.4.2 LDA: An Extreme Case of Three-way Metric Scaling

In contrary to the PCA where the n samples of X are considered from one entity, LDA treats the n samples from a number of classes or categories. In the training phase, along with the given X , the label for each sample is also available. For instance, if the training samples are from m classes, and each class has N samples², then $n = m \times N$.

LDA computes two types of variance according to the label of samples, the between-class variance S_B and the within-class variance S_W , then apply the Fisher criterion illustrated in Section 3.3 to gain the highest separability

²The number of samples from each class is not necessarily to be the same; for simplicity, here it is chosen to be the same.

between classes. Also noted that the sum of S_B and S_W is a constant that equals to the total variance S_T , if the training data are fixed.

Finally, LDA solves a linear equation, Eq 3.3. Since there is a fraction S_W^{-1} in the front, the equation can be further simplified to

$$S_B \omega = \lambda \omega \quad \text{subject to} \quad S_W = 1 \quad (3.11)$$

To make the S_W unity, a sphering (or whitening) process [48] is applicable and straightforward, which can be a preprocess. And three techniques as discussed in [48] can be applied directly. After sphering (or whitening), S_W is made to be unity. Noted that if S_W is unity, any unity column vector w can guarantee $\omega^T S_W \omega = 1$, as a result, Eq. 3.11 conforms to Eq. 3.3, which is the classical equation of LDA.

Recall the three-way scaling in Section 2.2.3, one extreme way to accommodate m subjects of n number of observations is to perform MDS on m number of averages derived from the n number of observations. In other words, given n number of $1 \times v$ samples, and knowing that they are from m classes with label information for each training samples, one extreme way of three-way scaling is to form a $m \times v$ matrix with each row filled by the respective mean first, and then to carry out classical embedding on this $m \times v$ matrix.

If the $m \times v$ matrix X_B has been obtained, LDA eigendecomposes $S_B = X_B^T X_B$, and the extreme three-way metric scaling eigendecomposes $X_B X_B^T$. The relationship between these two methods are exactly the same as the duality between PCA and metric MDS as stated in the previous section. Noted that LDA satisfies a condition that $\omega^T S_W \omega = 1$, so a sphereing (or whitening) preprocess is used to assure this.

3.4.2.1 Implementation of Embedded Version of LDA

To implement LDA as an embedding process, there are two steps. In step 1, it firstly extracts g number of class means, $\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_g$, from the given training samples $\mathbf{X}$ and label information t . Then a $g \times g$ matrix $\mathbf{D}^2$ is worked out by filling the element D_{ij}^2 with the squared Mahalanobis distances from $\mathbf{m}_i$ to $\mathbf{m}_j$.

$$D_{ij}^2 = (\mathbf{m}_i - \mathbf{m}_j) S_W^{-1} (\mathbf{m}_i - \mathbf{m}_j)' \quad (3.12)$$

where S_W^{-1} is the inverse of the within-class scatter matrix S_W of the whole data set $\mathbf{X}$. Noted that sometimes, inverting S_W is ill-posed. To make the inverting work, slightly adding a relatively small diagonal positive number to S_W will regularize the problem [49].

In step 2, since a squared distance matrix $\mathbf{D}^2$ has been obtained, metric scaling embedding described in Section 2.3.2.2 can be applied directly. After the embedding, a new configuration $\mathbf{Y}$ of size $g \times (g - 1)$ can be easily worked out denoting g class means embedded in a $(g - 1)$ -dimensional space. Of course, in such a case, we have assumed that all the class means are independent from one another.

3.5 Projecting an Instance into The Space

Both PCA and LDA simply project an instance into the space spanned by the principal components using matrix multiplication. In such a case, some information may be hidden. For example, along a perpendicular line to a plane,

we have two points with different heights to the plane. In such a case, a simple projection will definitely fail, because these two can never be differentiated if they are both projected onto the plane.

Take the embedded version of LDA as an example, from the g class means, $\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_g$, we obtain a metric space with $g - 1$ bases, assume that all the class means are mutually linear independent. For an arbitrary instance, we can project it into this space according to the distances to all those known class means, and the distances from all the class means to the origin in the original space, due to Gower [50]. Cuadras [51] stated that, so far, there seemed still no other alternative to perform such distance based projection. We first work out a g -dimensional vector $\mathbf{d}_0$, which is filled with all squared Mahalanobis distances from all class means to the origin in the original space.

Given an instance, $\mathbf{x} \in \mathbb{R}^v$, we work out another g -dimensional vector $\mathbf{d}$, which contains the squared Mahalanobis distances from $\mathbf{x}$ to each class mean. According to [50], we can have the ordinates $\mathbf{x}$ in the $(g - 1)$ -dimensional space as $\mathbf{y}$.

$$\mathbf{y} = -\frac{1}{2}(Y^T Y)^{-1} Y^T (\mathbf{d} - \mathbf{d}_0) \quad (3.13)$$

where Y is the final configuration of g number of class means in this case, i.e., Y is a $g \times (g - 1)$ matrix with each row a row vector denoting a class mean in the new space. This $(g - 1)$ -dimensional vector $\mathbf{y}$ can be considered as the extracted feature of the corresponding instance $\mathbf{x}$ in the metric space.

3.6 Discussions

Hotelling [43] described PCA as a technique of finding the major axes of an ellipsoid derived from a multivariate normal distribution. Following that most texts (eg. Anderson [52] Johnson and Wichern [53]) on classical multivariate analysis also present this topic following this fashion. However, most recently, some data analysts (e.g. Gower [54] and De Leeuw [55]) have taken an entirely different view on the PCA technique. They did not make any assumption about distributions³ or underlying statistical model. Rather, they use it as a technique for describing a set of data where certain algebraic or geometric optimization criteria are satisfied. This idea is actually very close to the original idea outlined by Pearson [5] in 1901, which is demonstrated in Section 3.4.1.1.

Applying embedding techniques, in this chapter I discussed the possibility to link the PCA, LDA and the Metric MDS methods. Through the analysis, at least there are three points worthy some attention in future.

First, there is a duality between the PCA and Metric MDS. Made it more precise, PCA is a special Metric MDS with the dissimilarity metric selected to be Euclidean. As a result, all the dimensionality reduction problems that can be achieved by PCA, now can also be achieved by Metric MDS. In addition, the dissimilarity metric used in Metric MDS can be tuned, by which, we may reveal some nonlinear structure of the data. In the next chapter, I will discuss some kernel generated metrics for the MDS to obtain nonlinear features.

Second, LDA can be considered as a case where all the class means are embedded using Metric MDS, subject to the data have been presphered using

³Noted that distance metric is most likely not distribution free, which means that when some geometric criteria are proposed, the underlying distribution has been implicitly, although not explicitly, assumed.

S_W . Alternatively, we can explain the LDA working mechanism this way. LDA embeds all the class means using Metric MDS, while the dissimilarity metric is chosen to be the Mahalanobis distance.

Third, LDA is very successful in some discriminant analysis cases. Only through the embedding of the class means, LDA still keeps most of the discriminant information. Meanwhile, in terms of computation, it demands much less computer power than the PCA, simply because PCA needs to embed all the samples at once. Based on this understanding, can we embed the class means but still keeping most of the variance of all samples? In 2002, Zhang. et al. [56] proposed a face recognition method to train eigenface by using only portion of the training samples to reduce the computation cost. In a special case, only one representative is selected from each class, the class mean, to train the eigenface.

Chapter 4

Kernel Generated Metric for Embedding

4.1 Introduction to Kernel Learning

Support vector learning [57][58][59][12][60] is also known as kernel learning, the reason is probably in that the success of support vector learning is highly based on a technique called the *kernel trick*. According to Vapnik, the data in a sparse space will be more probable to be linearly separable than in a dense space. In kernel learning, a data point x will be represented in a feature space F by a mapping $\phi(x)$, such that $\phi(x) = (\phi_1(x), \phi_2(x), \dots, \phi_{N_F}(x))$, where N_F is the dimension of the feature space. Note that N_F is generally huge, sometimes infinite; although it gains advantage of higher linear separability, the explicit mapping of every instance from the original space to the feature space becomes intractable.

One application of support vector learning is the support vector machine,

Table 4.1: Example of mercer kernel functions

$k(x_i, x_j)$	Name
$(x_i x_j + 1)^d$	Polynomial of order d
$\exp(-\frac{\ x_i - x_j\ ^2}{\delta})$	Gaussian (radial basis function)
$\exp(-\frac{\ x_i - x_j\ }{\delta})$	Exponential (radial basis function)
$\tanh(ax_i^T x_j + b)$	Sigmoid

which is a binary classifier based on the inner product in the feature space. So there comes a hope that we need not project all the data points to the feature space explicitly; instead, we may find the dot product of every pair in the feature space. In other words, we need not to work out the ϕ and every $x \mapsto \phi(x)$, rather, we just need to find all pairs of dot products $\langle \phi_i(x), \phi_j(x) \rangle$ for $1 \leq i \leq j \leq n$, where n is the number of instances.

If there is a function $k(x_i, x_j)$ such that $\langle \phi_i(x), \phi_j(x) \rangle = k(x_i, x_j)$ is always valid, the problem will be solved. In mathematics, the solution has been there for quite a while [61]. However, it is rediscovered by the learning community only recently [62][12]. The function k is so called kernel function, and a family of functions which satisfy the conditions are called mercer kernels, or the positive definite kernels. Some popularly used mercer kernel functions are listed in Table 4.1.

Through a mercer kernel function, all pairs of dot products in the feature space can always be worked out by a direct manipulation of the data x in the original space. This phenomenon is referred to as the kernel trick. At the same time, through the kernel trick, we can implicitly project all the data points into a very high dimensional feature space, and gain higher separability.

4.2 Kernel PCA and Its Embedding Interpretation

As mentioned in the previous chapter, PCA is a classical technique for extracting the structure from a set of data, possibly with a high dimensionality. And as a dimensionality reduction technique, PCA is efficient and powerful, also has a wide range of applications. Meanwhile, as we can see that in the kernel generated feature space, the dimension N_F is very high, or even sometimes infinity. Reducing the dimensionality of the data in the feature space naturally would attract some attention. At the same time, since the kernel mapping is generally nonlinear, the linear structure found in the feature space indeed reflects nonlinear structure in the original space. In other words, a set of nonlinear features can be obtained through PCA in the feature space. In 1998, Schölkopf et al. [14] reported their approach to generalize the PCA into the feature space domain. There are at least two advantages of Kernel PCA compared to the normal PCA.

1. Through a simple linear calculation (no nonlinear optimization) in the feature space, we can easily obtain the nonlinear components in the original space.
2. For some cases, we have a large number of training samples n , while the number of features in the original space, say v , is very small. In normal PCA, the number of components is confined by the $\min(n, v)$, while in kernel PCA, the number of components can be up to n , so if v is too small, we can still select more components to describe the data.

PCA is a technique that explains the variance-covariance structure of a set of variables through a few linear combinations of some principal components. However, for some underlying nonlinear data structures, PCA will fail. To deal with such cases, nonlinear statistical analysis models can be applied, but normally with very heavy computation loads for commonly used recursive algorithms. Kernel PCA can find some underlying nonlinear structures by simply applying the normal PCA on the feature space instead. For the second advantage, the number of principal components of normal PCA is governed by the smaller number between the number of samples and the dimension of each data sample. There are some cases where the dimension of the data sample is too small, so the number of components will be very small. However, in KPCA, because all the data samples have been (implicitly) mapped to a very high dimensional feature space, the number of components can be as high as the number of training samples, which may help to describe those samples better especially when the number of samples is much larger than the dimension of the data sets.

4.2.1 PCA in Feature Space

As in the support vector machine, the kernel PCA is heavily dependent on the kernel trick; that is, although normal PCA is conducted in the feature space, the algorithm needs to be based on the dot products $\langle \phi(x_i), \phi(x_j) \rangle$ exclusively in the feature space instead of the $\phi(x)$ s themselves.

From Eq. 3.1, we can see that PCA simply solves an eigenvalue equation with the covariance matrix (sometimes, also referred to as the total scatter matrix). In the feature space, kernel PCA also solves the same equation except

that the covariance matrix is derived from the feature data $\phi(x)$ instead of the x . Before going further, I would like to prove a property given in [14] that the eigenvectors will always lie in the span of the training samples, which plays a key role in converting an eigenvalue equation to a dot product based equation.

Theorem 1. [14] *Given a set of data, $x_1, x_2, \dots, x_n$, the eigenvector ω derived from the data set must lie in the span of the set of data themselves. In other words, if $\frac{1}{n} \sum_{i=1}^n x_i x_i^T \omega = \lambda \omega$, then $\omega = \sum_{i=1}^n \alpha_i x_i$ must hold, where α_i is some scalar parameter.*

Proof. $\therefore$

$$\begin{aligned}
 (xx^T)\omega &= \begin{pmatrix} x_1x_1 & x_1x_2 & \cdots & x_1x_n \\ x_2x_1 & x_2x_2 & \cdots & x_2x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_nx_1 & x_nx_2 & \cdots & x_nx_n \end{pmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{pmatrix} \\
 &= \begin{pmatrix} x_1x_1\omega_1 + x_1x_2\omega_2 + \cdots + x_1x_n\omega_n \\ x_2x_1\omega_1 + x_2x_2\omega_2 + \cdots + x_2x_n\omega_n \\ \vdots \\ x_nx_1\omega_1 + x_nx_2\omega_2 + \cdots + x_nx_n\omega_n \end{pmatrix} \\
 &= \begin{pmatrix} (x_1\omega_1 + x_2\omega_2 + \cdots + x_n\omega_n)x_1 \\ (x_1\omega_1 + x_2\omega_2 + \cdots + x_n\omega_n)x_2 \\ \vdots \\ (x_1\omega_1 + x_2\omega_2 + \cdots + x_n\omega_n)x_n \end{pmatrix} \\
 &= (x_1\omega_1 + x_2\omega_2 + \cdots + x_n\omega_n) \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \\
 &= (x \cdot \omega)x
 \end{aligned} \tag{4.1}$$

and

$$\frac{1}{n} \sum_{i=1}^n x_i x_i^T \omega = \lambda \omega \tag{4.2}$$

$$\begin{aligned}
\omega &= \frac{1}{n\lambda} \sum_{i=1}^n x_i x_i^T \omega \\
&= \frac{1}{n\lambda} \sum_{i=1}^n (x_i \cdot \omega) x_i \\
&= \sum_{i=1}^n \alpha_i x_i
\end{aligned} \tag{4.3}$$

It can be seen that the above can be quite trivial by recognizing that the term $x_i^T \omega$ in Eq. 4.2 and Eq. 4.3 is a scalar.

In the feature space $\mathcal{F}$, all the x have been mapped to $\phi(x)$, and the covariance matrix becomes $S_T = \frac{1}{n} \sum_{i=1}^n \phi(x_i) \phi(x_i)^T$, and according to the proof,

$$\omega = \sum_{i=1}^n \alpha_i \phi(x_i) \tag{4.4}$$

and

$$\lambda \omega = \frac{1}{n} \sum_{i=1}^n \phi(x_i) \phi(x_i)^T \omega \tag{4.5}$$

dot multiply both RHS and LHS of Eq. 4.5 with $\phi(x_k)$ for all $k = 1, 2, \dots, n$, total number of n equations are obtained.

$$\lambda \phi(x_k) \cdot \omega = \frac{1}{n} \phi(x_k) \cdot \left(\sum_{i=1}^n \phi(x_i) \phi(x_i)^T \omega \right) \quad \text{for all } k = 1, 2, \dots, n \tag{4.6}$$

Here, we have assumed that all the $\phi(x)$ have been properly centered in the feature space, for detailed algorithm to do centering in feature space, the reader can refer to the Appendix B in [14]

Substitute Eq. 4.4 into Eq. 4.6, we get

$$\lambda \sum_{i=1}^n \alpha_i (\phi(x_k) \cdot \alpha(x_i)) = \frac{1}{n} \sum_{i=1}^n \alpha_i (\phi(x_k) \cdot \sum_{j=1}^n \phi(x_j) (\phi(x_j) \cdot \phi(x_i))) \text{ for all } k = 1, 2, \dots, n \quad (4.7)$$

If an $n \times n$ kernel matrix K is defined, with ij -th element is the dot product kernel function $k(x_i, x_j)$, i.e., $K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle$, Eq. 4.7 can be expressed in a matrix form as

$$n\lambda K\alpha = K^2\alpha \quad (4.8)$$

and Eq. 4.8 can be further simplified to

$$n\lambda\alpha = K\alpha \quad (4.9)$$

where α is a $n \times 1$ vector, whose i -th element is α_i .

Just as a normal PCA, a set of eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ and their corresponding eigenvectors $\alpha_1, \alpha_2, \dots, \alpha_n$ are obtained. We select p eigenvectors to form a kernel components space, V^p , where $p \leq n$. Whenever there comes a new test data vector x , the projection is straightforward. First, the x is mapped into the feature space, and $\phi(x)$ is obtained. Then, the projection to the kernel component space will be

$$\langle V^p, \phi(x) \rangle = \sum_{i=1}^p \alpha_i^p \langle x_i, x \rangle, \quad 1 \leq p \leq n \quad (4.10)$$

4.2.2 Embedding Interpretation of KPCA

Kernel PCA finds the kernel principal components by eigendecomposing a centered kernel matrix K (Eq. 4.9), and it is well-known that the inner product could be considered as a kind of similarity measure. Intuitively, it may be possible that kernel PCA can be interpreted in an embedding context. Williams [22] reported his findings on the connection between the kernel PCA and metric MDS recently.

Since the embedding techniques are based on the preservation of the pairwise distances and the kernel trick solely based on the dot product in the feature space, the relationship between dot product (similarity) and distance (dissimilarity) surely will play a crucial role.

The squared Euclidean distance between two points, $\phi(x_i)$ and $\phi(x_j)$, in feature space can be simply denoted as dot product terms as shown below [16] [18].

$$\begin{aligned}
 d_{ij}^2 &= \|\phi(x_i) - \phi(x_j)\|^2 \\
 &= \phi(x_i)^T \phi(x_i) + \phi(x_j)^T \phi(x_j) - 2\phi(x_i)^T \phi(x_j) \\
 &= k(x_i, x_i) + k(x_j, x_j) - 2k(x_i, x_j)
 \end{aligned} \tag{4.11}$$

where $k(\cdot)$ is the kernel function playing the kernel trick. In other words, the distances between two feature points can be worked out without explicitly obtaining the feature points themselves first. The kernel trick can be directly applied to the distance [16].

As we can see from the section 2.3.2.2, metric MDS is exclusively based

on an $n \times n$ square distance matrix D^2 , where the ij -th element is the square distance between i and j instances. A further step from Eq. 4.11, we obtain the matrix D^2 easily,

$$D^2 = \mathbf{k}\mathbf{1}_n^T + \mathbf{1}_n\mathbf{k}^T - 2K \quad (4.12)$$

where K is the kernel matrix, $\mathbf{1}_n$ is a $n \times 1$ vector filled with all 1s, and $\mathbf{k}$ is another $n \times 1$ vector filled with all diagonal elements from K .

Since metric MDS eigendecomposes $\mathbf{B} = -\frac{1}{2}HD^2H$, and $\mathbf{B}$ is the double centered squared distance matrix. In the feature space, the squared distance matrix is denoted as shown in Eq. 4.12. To obtain a metric MDS for the feature data points, just simply substitute the D^2 in Eq. 4.12 to $\mathbf{B}$, and then eigendecompose the term $\mathbf{B}$.

$$\begin{aligned} \mathbf{B} &= -\frac{1}{2}H(\mathbf{k}\mathbf{1}_n^T + \mathbf{1}_n\mathbf{k}^T - 2K)H \\ &= HKH - H(\frac{1}{2}\mathbf{k}\mathbf{1}_n^T)H - H(\frac{1}{2}\mathbf{1}_n\mathbf{k}^T)H \\ &= HKH - H\frac{1}{2}\mathbf{k}\mathbf{0}_n^T - \mathbf{0}_n\frac{1}{2}\mathbf{k}^TH \\ &= HKH \end{aligned} \quad (4.13)$$

where $\mathbf{0}_n$ is a $n \times 1$ vector filled with all 0s. We know that the centering matrix $H = I - \frac{\mathbf{1}_n\mathbf{1}_n^T}{n}$, or H is a $n \times n$ matrix with each diagonal element $\frac{n-1}{n}$, and all others $-\frac{1}{n}$. The meaning of doing $\mathbf{1}_n^TH$ is to obtain a $1 \times n$ row vector with each element the summation of all elements from the respective column in the matrix H . Take the 0_1 as an example, it equals to the summation from all the elements of the first column of H , i.e., $H_{11} + H_{21} + \dots + H_{n1}$, which is clearly

0 because $H_{11} = \frac{n-1}{n}$ and all other $n-1$ elements equals to $-\frac{1}{n}$. As such, in the above equation, $\mathbf{1}_n^T H = \mathbf{0}_n^T$ and $H \mathbf{1}_n = \mathbf{0}_n$.

Eq. 4.9 shows that Kernel PCA obtains the so called Kernel principal components by eigendecomposing the kernel matrix K . From a metric MDS point of view, the derived matrix B is HKH , which is the centered kernel matrix K , then a further eigendecomposition may be applied to obtain the so called principal coordinates. Noted that in the derivation of Eq. 4.9, I mentioned that there should be a preprocess to centering the data in the feature space, so effectively, both approaches obtain the Kernel principal components (or coordinates) by eigendecomposing the centered kernel matrix K . In other words, the Kernel PCA can be interpreted as follows in the context of metric MDS. Kernel PCA firstly projects all the data points x to a very high dimensional space; after that, all the pairwise squared Euclidean distances, $d_{i,j}^2$, between every two feature points, $\phi(x_i)$ and $\phi(x_j)$, are obtained, Lastly, a metric MDS is applied to obtain the principal components.

4.3 Kernel LDA

Since PCA has been successfully generalized to the feature space, LDA would naturally be generalized to the feature space. Kernel LDA [63] solves the linear discriminant analysis problem in a kernel feature space, such that in the input space, the discriminant components can be nonlinear. In [63], however, only a classical two-class discriminant case was proposed. Baudat and Anouar [15] further generalized the two-class case to a multi-class discriminant problem, and named the technique Generalized Discriminant Analysis (GDA). In this thesis, I

will only review the more generalized multi-class case, but still I prefer to name it kernel LDA.

The same as a normal LDA, the kernel LDA targets at maximizing a term known as the Fisher's criterion, which is a ratio of the between-class and within-class variance, $\frac{\omega^T S_B^\Phi \omega}{\omega^T S_W^\Phi \omega}$. Compared to the normal LDA, the only difference is that the vector w , the between-class variance S_B^Φ and the within-class variance S_W^Φ are all based on the feature data points. Since the dimension of feature space $\mathcal{F}$ is too high, any direct implementation on the feature vector itself will be infeasible. Again, there comes the kernel trick, which means the fisher criterion shall use only the dot product $\langle \phi(x_i), \phi(x_j) \rangle$ exclusively.

Just as what have been done in normal LDA, the Lagrange multiplier method is utilized on maximizing the Fisher's criterion, and it comes to a canonical form,

$$S_B^\Phi \omega = \lambda S_W^\Phi \omega \quad (4.14)$$

Multiply both sides of Eq. 4.14 with a common factor $\phi(x_{ij})'$, the above equation becomes

$$\phi(x_{ij})' S_B \omega = \lambda \phi(x_{ij})' S_W^\Phi \omega \quad (4.15)$$

Let g be the number of classes, n is the total number of data points, and n_l is the number of data points of class l , S_W and S_B become

$$S_W = \frac{1}{n} \sum_{l=1}^g \sum_{k=1}^{n_l} \phi(x_{lk}) \phi(x_{lk})'$$

$$S_B = \frac{1}{n} \sum_{l=1}^g n_l \bar{\phi}_l \bar{\phi}_l' \quad \text{where} \quad \bar{\phi} = \frac{1}{n_l} \sum_{k=1}^{n_l} \phi(x_{lk})$$

Recall Eq. 4.4, the eigenvector can be expressed as a combination of all the kernel features, and this equation can be further modified here as

$$\omega = \sum_{p=1}^g \sum_{q=1}^{n_p} \alpha_{pq} \phi(x_{pq}) \quad (4.16)$$

The RHS of Eq. 4.15 can be modified to

$$\begin{aligned} S_W \omega &= \frac{1}{n} \sum_{l=1}^g \sum_{k=1}^{n_l} \phi(x_{lk}) \phi(x_{lk})' \sum_{p=1}^g \sum_{q=1}^{n_p} \alpha_{pq} \phi(x_{pq}) \\ &= \frac{1}{n} \sum_{p=1}^g \sum_{q=1}^{n_p} \alpha_{pq} \sum_{l=1}^g \sum_{k=1}^{n_l} \phi(x_{lk}) [\phi(x_{lk})' \phi(x_{pq})] \\ \lambda \phi(x_{ij})' S_W^\Phi \omega &= \frac{\lambda}{n} \sum_{p=1}^g \sum_{q=1}^{n_p} \alpha_{pq} \sum_{l=1}^g \sum_{k=1}^{n_l} [\phi(x_{ij})' \phi(x_{lk})] [\phi(x_{lk})' \phi(x_{pq})] \\ &= \frac{\lambda}{n} K K \alpha \end{aligned} \quad (4.17)$$

Similarly, the LHS of Eq. 4.15 can be modified to

$$\begin{aligned} S_B \omega &= \frac{1}{n} \sum_{p=1}^g \sum_{q=1}^{n_q} \alpha_{pq} \sum_{l=1}^g \left[\sum_{k=1}^{n_l} \phi(x_{lk}) \right] \left[\frac{1}{n_l} \right] \left[\sum_{k=1}^{n_l} \phi(x_{lk})' \phi(x_{pq}) \right] \\ \phi(x_{ij})' S_B \omega &= \frac{1}{n} \sum_{p=1}^g \sum_{q=1}^{n_q} \alpha_{pq} \sum_{l=1}^g \left[\sum_{k=1}^{n_l} \phi(x_{ij})' \phi(x_{lk}) \right] \left[\frac{1}{n_l} \right] \left[\sum_{k=1}^{n_l} \phi(x_{lk})' \phi(x_{pq}) \right] \\ &= \frac{1}{n} K W K \alpha \end{aligned} \quad (4.18)$$

Combine Eq. 4.17 and Eq. 4.18, the equation becomes

$$\alpha^T K W K \alpha = \lambda \alpha^T K K \alpha \quad (4.19)$$

where, K is the kernel matrix of size $n \times n$. W is a block diagonal matrix of size $n \times n$, where i -th block is filled by a constant $\frac{1}{n_i}$, and n_i is the number of instances in the class i . In addition, all other elements of W will be filled with 0's.

4.4 Convolution Kernel Generated Metric for Embedding

In a general pattern classification setting, we are given a set of training data $\mathcal{D} \subseteq X \times \mathcal{T}$ of n instances $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \dots, (\mathbf{x}_n, t_n)\}$, where X denotes the data space and all g possible classes are labelled by $\mathcal{T}$. Kernel PCA is an unsupervised feature extraction process, it does not incorporate the class information at all. Intuitively, the given class information is crucial in extracting the discriminant features. Kernel LDA utilized this information implicitly in the kernel space, and due to the implicity, it is difficult to view it from an embedding point of view. Zhang et al. [18] [23] reported their approach where a discriminant kernel function is achieved through a combination of the data space and the class information in a single kernel function. Convolution kernels from Haussler [64] play key roles in combining the X and $\mathcal{T}$ in one kernel function.

4.4.1 Tensor Product and Direct Sum Kernels

There are a number of different methods to construct new kernels from existing ones [65]. Two techniques based on convolution kernel are utilized here, tensor product and direct sum. As detailed in [56], given $x_1, x_2 \in X$ and $t_1, t_2 \in T$. If k_1, k_2 are (positive definite) kernels defined on $X \times X$ and $T \times T$ respectively, then the tensor product kernel², $k_1 \otimes k_2$, is defined on $(X \times T) \times (X \times T)$ as

$$k_1 \otimes k_2((x_1, t_1), (x_2, t_2)) = k_1(x_1, x_2)k_2(t_1, t_2) \quad (4.20)$$

Similarly, the direct sum kernel, $k_1 \oplus k_2$, defined on $(X \times T) \times (X \times T)$ as

$$k_1 \oplus k_2((x_1, t_1), (x_2, t_2)) = k_1(x_1, x_2) + k_2(t_1, t_2) \quad (4.21)$$

Both tensor product and direct sum kernels are proved to be positive definite kernels in [64] and [65], so all the tricks played on a normal kernel function can also be applied to these two convolution kernel functions.

4.4.2 Construction of Distance Matrix

As mentioned above, we are given both the data and the class label information for training in general pattern classification problems, and the tensor product and direct sum kernels above make it possible of combining these two parts together in generating a so called discriminant distance matrix.

Tensor Product Kernel is derived from the mathematical term "Vector Space Tensor Product", which is not directly related to mathematical term "Tensor" if there are possibly some relationships. Vector Space Tensor Product defines a way to create a new vector space from two vector spaces. Please referred to <http://mathworld.wolfram.com/VectorSpaceTensorProduct.html>

We choose kernels following the authors of [18]. For k_1 , we choose the commonly used Gaussian kernel.

$$k_1(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\delta}\right)$$

where $\delta > 0$ is a scaling constant. And Linear kernel is chosen for k_2 .

$$k_2(t_i, t_j) = t_i' t_j$$

Recall Eq. 4.20 and Eq. 4.21 above, the tensor product and direct sum kernels become

$$(k_1 \otimes k_2)_{ij} = \begin{cases} \exp\left(-\frac{\|x_i - x_j\|^2}{\delta}\right) & \text{if } t_i = t_j \\ 0 & \text{if } t_i \neq t_j \end{cases} \quad (4.22)$$

and

$$(k_1 \oplus k_2)_{ij} = \begin{cases} \exp\left(-\frac{\|x_i - x_j\|^2}{\delta}\right) + 1 & \text{if } t_i = t_j \\ \exp\left(-\frac{\|x_i - x_j\|^2}{\delta}\right) & \text{if } t_i \neq t_j \end{cases} \quad (4.23)$$

respectively.

According to the kernel trick for distance in Eq. 4.11, the squared distances between any two feature points in the tensor product and direct sum kernels generated feature spaces can be worked out. For tensor product kernel, we have

$$d_{ij}^2 = \begin{cases} 2 - 2 \exp\left(-\frac{\|x_i - x_j\|^2}{\delta}\right) & \text{if } t_i = t_j \\ 2 & \text{if } t_i \neq t_j \end{cases} \quad (4.24)$$

while for direct sum kernel, we have

$$d_{ij}^2 = \begin{cases} 2 - 2 \exp(-\frac{\|x_i - x_j\|^2}{\delta}) & \text{if } t_i = t_j \\ 4 - 2 \exp(-\frac{\|x_i - x_j\|^2}{\delta}) & \text{if } t_i \neq t_j \end{cases} \quad (4.25)$$

Eq. 4.24 and Eq. 4.25 reveal a nice distance property for tensor product and direct sum kernels. Under these two kernel mappings, in the feature space, inter-class and intra-class distances can be easily distinguished with a certain threshold. Fig. 4.1 from [18] illustrates the property straightforwardly.

Figure 4.1: Relationships between the Intra-class and Inter-class distances in Tensor Product and Direct Sum Mapped Feature Space

Based on the observation, a distance matrix can be constructed that contains both the pattern and label information. In that distance matrix, a single distance metric is given, and with the class information, the discriminant power will be improved. Fig. 4.1 also shows that the direct sum kernel will provide larger hysteresis, i.e., the higher possibility of separability, so the direct sum kernel will be used in experiments in this thesis.

4.5 Experiments

In this section, a set of experiments are conducted to verify that the embedded version of KPCA performs very closely compared to the original KPCA in pattern classification. At the same time, the KLDA and embedding of convolutional kernel generated distance metric are also applied on the same set of data sets. Some observations and reasonings will also be discussed in this chapter regarding to the recognition results.

4.5.1 Experimental Settings and Results

In experiments, we select six data sets from UCI machine learning repository [66], and perform four sets of classification experiments. The six data sets are

1. Ionosphere: Number of instances $N = 351$, Number of classes $g = 2$, and Dimensionality $d = 34$
2. Iris: Number of instances $N = 150$, Number of classes $g = 3$, and Dimensionality $d = 4$
3. Wine: Number of instances $N = 168$, Number of classes $g = 3$, and Dimensionality $d = 12$
4. Balance: Number of instances $N = 625$, Number of classes $g = 3$, and Dimensionality $d = 4$
5. Breast Cancer: Number of instances $N = 569$, Number of classes $g = 2$, and Dimensionality $d = 30$

6. Soy Bean: Number of instances $N = 47$, Number of classes $g = 4$, and Dimensionality $d = 8$

And the four sets of experiments are based on kernel PCA, embedded version of kernel PCA, kernel LDA and the convolution kernel classification.

In all the experiments, we randomly select about 20% from each class as the training samples, and the rest will be the testing instances. Take Iris data set as an example, it comprises 3 classes of samples, and each class has 50 samples, so totally the data set comprises 150 samples. A random selection program is written to randomly select 30 (or 20%) samples from the pool of 150 samples, and each training sample is known from which class. After the data samples have been partitioned into training samples and testing one, a mapping spaces is obtained based on these training samples using a certain technique. Because the class labels of training samples are known, it is straightforward to work out the mean of each class in the obtained space. Finally, every test sample is mapped into the obtained space, and a simple nearest mean process can be applied to obtain the class label. All four methods are conducted on the data, and the recognition accuracies are recorded. Finally, the same process, from random sampling till conducting recognition, will be repeated for 100 times for cross validations, and the average accuracy and the deviation will be reported. Table 4.2 lists the detailed average accuracy of each method on each data set, meanwhile Fig. 4.2 diagrammatically shows the error bars in addition to the average accuracies.

Table 4.2: Average recognition accuracy for six UCI data sets

	KPCA	Embedded KPCA	KLDA	Convolution Kernel
Ionosphere	75.9%	75.6%	92.3%	78.7%
Iris	92.0%	92.8%	95.3%	93.2%
Wine	71.8%	71.2%	71.7%	94.4%
Balance	76.2%	75.3%	77.6%	73.6%
Breast Cancer	90.5%	91.4%	92.5%	99.9%
Soy Bean	90.0%	90.8%	95.4%	97.6%

4.5.2 Observations and Reasoning

The best classification accuracy for each data set has been bolded in Table 4.2; also it is clearly shown in the Fig. 4.2, on top of that the error bars are indicated in the drawing too. There are three observations listed below, and the reasoning is in the next three paragraphs, respectively.

1. Kernel PCA and its embedded version perform almost the same, but with some fluctuations.
2. Convolutional kernel embedding methods performs better than the kernel PCA and its embedded version.
3. Generally, kernel LDA and convolution kernel embedding perform better than the kernel PCA, however kernel LDA performs worse on 3 data sets than the convolution kernel embedding.

The kernel PCA and its embedded version generate close classification accuracies, which is understandable. Both methods indeed eigendecompose a same double centered kernel matrix $\mathbf{K}$ as shown in Eq. 4.9 and Eq. 4.13 respectively. As such, the two subspaces are spanned by the same set of basis vectors. The only difference between these two methods lies in the projection. Kernel PCA

Figure 4.2: Classification accuracy on six UCI data sets. In each panel, from left to right, the four bars record the KPCA, Embedded KPCA, KLDA, and convolution kernel embedding, respectively.

projects a new instance to the subspace according Eq. 4.10. In such case, the dot product between this instance in the feature space with every training Samples in the feature space will be carried out first. Then the dot product terms are summed up with the known weight factors to obtain the coordinates in the subspace. However, in embedded version of KPCA, as shown in Eq. 3.2, from the original instance matrix and the obtained subspace, we first work out a mapping matrix. Then the new instance will be directly mapped into the subspace by multiplying the mapping matrix. Since the success of both KPCA and its

embedded version mainly relies on the subspace that learned, the classification accuracies will be very similar between these two methods.

In all experiments, the kernel function is chosen to be the Gaussian radial basis function. According to Eq. 4.11, the distance between two instances x_i and x_j can be formulated with only dot products $k(x_i, x_i) + k(x_j, x_j) - 2k(x_i, x_j)$, also noted that for Gaussian kernel, $k(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{\delta})$. Subsequently, the distance between two instances x_i and x_j will be $2 - 2k(x_i, x_j)$. On the other hand, the direct sum convolution kernel was utilized in the convolution kernel embedding in the experiments, based on the Eq. 4.25, we can find that the distances between two instances from the same class is governed by the same distance equation above, while the distances between instances from different classes need to add another 2 to the distance shown above. As a results, the distance metric used in the convolutional kernel embedding has higher separability than the one used in the kernel PCA and its embedded version.

Theoretically, kernel LDA shall be the best discriminant technique compared to the other three, however, it only performs best in Ionosphere and Balance data sets. The reason is that there are some assumptions of the data set, which may be crucial for the success of the kernel LDA. Unfortunately, sometimes, the assumptions are violated. Take Soy Bean data set as an example, the data set only comprises 47 instances, each described by 8 variables, but it is divided into 4 classes. In such a case, there is no sufficient amount of training samples to satisfy the requirement of the kernel LDA method. On the other hand, the convolution kernel embedding method is rather an ad-hoc method, it deliberately design two distance metrics for the training instances according to whether the two is from a same class or not, so there is no assumption to be

fulfilled except that the training samples are correctly labelled. For small sample cases, it will not be surprising to see that the performance of the convolution kernel embedding surpasses the kernel LDA.

4.6 Discussions

In this chapter, the kernel trick to distance is discussed followed by the application to an embedding process to derive a subspace. Under the framework of this thesis, this chapter presents some interesting distance metrics generated from the kernel trick.

Kernel learning methods are overwhelmingly based on a technique referred to as the kernel trick, which enables some feasible mathematical work in a very high dimensional space. The trick is played such that we do not need to project all the data to the very high dimensional feature space explicitly, only the dot product terms in the feature space are worked out, and the algorithm will stick to working around with dot product terms only.

Since the dot product can be considered as a kind of similarity measure, and distance is a special dissimilarity metric, all the kernel based algorithms can be interpreted in terms of analysis of distances. In a further step, many of the common kernel methods can be interpreted as an embedding process. Based on this thought, the kernel PCA is discussed in embedding context. Also a special distance metric which combines the data variable and the label information through the convolution kernel is also discussed.

Because the whitening process, as mentioned in section 3.4.2, for the data in the very high dimensional feature space is not that straightforward, the deriva-

tion from kernel LDA to its embedded version has not worked out at this moment. However, it is a promising future direction. The reason is that, if the whitening process in the feature space can be achieved in an effective way, there will be more conventional techniques that can be transferred to the kernel domain without any obstacle.

The embedding methods are solely based on the distance preservation criterion, and they reduce certain level of complexity of the classifier design because the classifier normally takes distance as the criterion in classification, notably the nearest neighbor method. On the contrary, for the variance analysis based methods, the classifier may be based on the more costly variance matching criterion such as maximum likelihood. Some assumptions are also made in the embedding methods, such as what distance metric is more suitable for a given data set, and these assumptions may be made either implicitly or explicitly. The embedding methods come with more example-based classifier context. And the foreseen problem of this kind of methods is obvious, the embedding methods take all the training samples as correct samples, if there are some false or noisy samples inside, the performance of the classifier will be severely worsened.

Chapter 5

Manifold Embedding: ISOMAP and LLE

5.1 Introduction

PCA and MDS are very popularly used techniques in dimensionality reduction. However, only when the underlying manifolds are linear or perturbations from such ones, PCA and metric MDS are suitable. For some globally nonlinear structured manifold, PCA and MDS will fail. Take a look at a prototype example, a segment of a one-dimensional manifold $\mathbf{M}$ in $\mathbb{R}^3$, as shown in Fig. 1.2 again. The goal of reducing the dimension for this data set is of course to find a one-dimensional space to embed all the data points. However, normal linear models will definitely fail. In terms of the embedding techniques we discussed in previous chapters, no matter how hard we tune the distance metric, we will still fail with great chances. The reason is simply that one single universal distance metric is no longer suitable for all pairs of the data points. The distance

between any two points along the manifold is a curved distance instead of a directly-linked straight distance as used in PCA or conventional metric MDS. This type of distance is referred to as geodesic distance in mathematics, and there are solutions towards finding such distances if the analytical form of the manifold is known. (Of course, the calculation may be tedious.) [21]. However, in most engineering applications, the analytical form may not be known and only a set of samples can be obtained from the manifold, finding an easy solution to reveal the structure of the manifold without reconstruct the analytical form of the manifold is of great importance.

Fortunately, according to the important Taylor's theorem, any differentiable function becomes approximately linear in a sufficiently small region around a point [21]. Based on this theorem, Tenenbaum et al. [7] proposed an approach called ISOMAP, and Roweis and Saul [8] proposed another method, named Local Linear Embedding (LLE). For a set of given samples, every two samples may be categorized as either neighboring samples or non-neighboring samples according to the distance between these two samples. According to the Taylor's theorem, two neighboring samples can be considered lying on a linear space, and Euclidean distance is sufficient and appropriate to address the closeness of these two points. But for non-neighboring points, since they are linked along a nonlinear curvature, the Euclidean distance may not be a right measurement of the closeness of the two samples.

ISOMAP firstly calculated all the Euclidean distances for neighboring points. Then, it utilized the Dijkstra's algorithm to find the shortest path for every two non-neighboring samples, and these distances are used to simulate the curved distances. Finally, it formed the squared distance matrix D^2 (ref to section

2.3.2.2) by filling all the squared distances obtained. Following that, a metric MDS is applied to the learned matrix D^2 to embed all the samples into a (possibly) lower dimensional space. In other words, ISOMAP defines the local and global distances in a different way (using different metrics), and then embed the derived distances to a single metric space using MDS. So ISOMAP preserves the global structure while keeping the local information. On the other hand, LLE embeds all the points according to topography-preserving instead of the pairwise distance-preserving. First, every point is represented as a weighted sum of all its neighbors (or local information) in the original data space. Then, during the embedding, all the weights for every neighboring point in the learned space shall be approximately the same to the correspondent weights to the original point. As a result, all the local (neighborhood) information is kept in the (possibly) lower dimensional space.

Before going deep into ISOMAP and LLE implementations, I would like to review a concept, latent variables, which could be the mathematical explanation of the reason why the dimensionality reduction is possible for some cases like the one in Fig. 1.2.

Latent Variables: Sometimes, a set of very high dimensional data are actually generated or governed by only a small number of variables. And these variables are sometimes referred as latent variables [67] [68].

Take the manifold $\mathbf{M}$ in Figure 1.2 as an example again, this $\mathbf{M}$ is actually parameterized by a single variable $t \in [0,1]$, and $\mathbf{M} = \{f(t) = [\sin 2\pi t, \cos 2\pi t, t]'\}$. It is clear that the latent variable is t , and the number of latent variables is one. In other words, the manifold $\mathbf{M}$ can be essentially embedded into a single dimensional space spanned by t . In a sequel, the dimension reduction problem

in this context is defined as a learning process to investigate the number of the latent variables for a set of data and then map all the data points into a space spanned by all those latent variables.

Face representation demonstrated in [7][8] also exemplified the applicability of the latent variable idea in practical ways. Every image appears as a very high dimensional data vector, for instance, in a computerized digital image format; even a 256 x 256 one costs total 65536 pixels to represent. However, according to Tenenbaum [7], two factors are far enough to represent all the face images for a single person, they are azimuth and elevation¹. In other words, the face is either rotated from left to right or from top to down. In [8], Roweis and Saul applied LLE to a face image database like the one used in [7], and derived a manifold of a single person's according to his pose and expression instead.

5.2 Performing ISOMAP

To perform ISOMAP, we need to go through three steps. The first two are designed to obtain a squared distance matrix D^2 , and the last step embeds all the training samples based on the D^2 using metric MDS as described in section 2.3.2.2. The first two steps try to work out all pairwise pseudo-geodesic distances for the training samples using the Dijkstra's algorithm. Given a set of samples, or data points in the space, and every two points are linked by an edge with known distance, Dijkstra's algorithm is capable to find out a chain of short paths between any two points. During the course, the nonlinear structure of the manifold, such as a segment of a spiral shown in Fig. 1.2, has been decoiled.

In future discussions from Tenenbaum (eg. in [20]), the lighting conditions are also considered as another factor.

And the following embedding process will find a (possibly) lower dimensional space to accommodate all the training samples while preserving the decoiled structure as much as possible.

Given a collection $\{x_i\}_{i=1}^n$ of training points, the algorithm to get the pairwise geodesic distances comprises two steps.

First, a neighborhood graph is constructed, and there are two ways to determine whether two points x_i and x_j are neighbors. Either we draw a circle at a point x_i with radius ϵ , and every point within the circle will be a neighbor to the x_i , this method is called ϵ -Isomap. Or we set a number K , such that K number of nearest points to x_i will be considered as neighbors to x_i , we call this method K -Isomap.

Second, we start to compute all the pairwise geodesic distances. For the neighboring points, the distance is just the Euclidean distance. But for the far-away points, the distance is calculated this way. Initially, we set the distance $d_{i,j} = \infty$. Then, we replace $d_{i,j}$ by $\min(d_{i,j}, d_{i,k} + d_{k,j})$ for each $k = 1, 2, \dots, n$.

Finally, we get all the pairwise geodesic distances, and naturally we can get the squared distance matrix D^2 . The metric MDS process discussed in Chapter 2 can be easily applied to get (possibly) lower dimensional representations for all the data sample points.

From the implementation, it is clear that ISOMAP can be categorized under the framework of this thesis. ISOMAP selects all the training data points as representatives, and uses a special distance metric, which is heterogenous. In other words, the neighboring data points has a different distance metric compared to those non-neighboring data points.

5.3 Performing LLE

"Think Globally, Fit Locally" [69], just as what the title indicated, LLE provides a way to find the global manifold through some investigations on the local linear relationships only. Given a set of $\{x_i\}_{i=1}^n$ training samples, three steps are required to derive a set of $\{y_i\}_{i=1}^n$ data points, where the dimension of y is smaller than x .

Step 1 is exactly the same as what we have done in ISOMAP to get the neighborhood graph.

In step 2, assume K-neighbors approach is adopted, we compute $\{w_{i1}, w_{i2}, \dots, w_{iK}\}_{i=1}^n$, or for each point from $\{x_1, x_2, \dots, x_n\}$, we compute K weights so that the point can be reconstructed from its K neighbors through a minimization of Eq. 5.1 subject to $\sum_{j \in \mathcal{N}(x_i)} w_{ij} = 1$. Since $\hat{x}_i = \sum_{j \in \mathcal{N}(x_i)} w_{ij} x_j$

$$\begin{aligned} S_1(W) &\triangleq \sum_{i=1}^n \|x_i - \hat{x}_i\|^2 \\ &= \sum_{i=1}^n \left\| x_i - \sum_{j \in \mathcal{N}(x_i)} w_{ij} x_j \right\|^2 \end{aligned} \quad (5.1)$$

where $\mathcal{N}(x_i)$ is the set of neighbors of point x_i . Ideally, minimizing S_1 can guarantee that the weights are reflecting the intrinsic local geometric property of the manifold.

To solve Eq. 5.1, let us consider a particular point x with K neighbors η_j and weights w_j that sum to one, so this particular cost function will be

$$s_1 = \left\| x - \sum_{j=1}^K w_j \eta_j \right\|^2 \quad (5.2)$$

It is easy to see that if every data point, including x and $\{\eta_1, \eta_2, \dots, \eta_k\}$, is rotated by the same angle or scaled by the same magnitude, the RHS of Eq. 5.2 will be multiplied by a constant. Meanwhile, in order to minimize Eq. 5.2, s_1 will be set as 0, so w will not be affected by that multiplied constant factor. As a result, the cost function Eq. 5.1 is invariant to rotation and uniform scaling of the neighborhoods. Because $\sum_{j=1}^K w_j = 1$ is enforced, above Eq. 5.2 can be rewritten as

$$s_1 = \left\| \sum_{j=1}^K w_j (x - \eta_j) \right\|^2 \quad (5.3)$$

From Eq. 5.3, we can also tell that w is invariant to translation, because if x and all the η 's are translated by a same magnitude, the RHS of Eq. 5.3 is kept unchanged.

If a local Gram matrix of size $K \times K$ is defined such that the jk -th component $G_{jk} = (x - \eta_j) \cdot (x - \eta_k)$, Eq. 5.3 can be transformed to

$$s_1 = \sum_j \sum_k w_j w_k G_{jk} \quad (5.4)$$

Because the local Gram matrix G is symmetric and semipositive definite, it is invertible. The loss function can be minimized analytically using a Lagrange multiplier with the enforcement that $\sum_j w_j = 1$. The optimal weight for j -th neighbor w_j is

$$w_j = \frac{\sum_k G_{jk}^{-1}}{\sum_l \sum_m G_{lm}^{-1}} \quad (5.5)$$

where the denominator is the summation of all the elements in the inverse of G .

Step 3 will preserve the weights obtained above in a lower dimensional space. For all the y_i , we can approximate it according to the weights $\{w_{ij}\}_{j=1}^K$, such that $\hat{y}_i = \sum_{j \in \mathcal{N}(x_i)} w_{ij} y_j$, and then we minimize a second least square cost function S_2 defined as

$$\begin{aligned} S_2(\{y_i\}_{i=1}^n) &\triangleq \sum_{i=1}^n \|y_i - \hat{y}_i\|^2 \\ &= \sum_{i=1}^n \|y_i - \sum_{j \in \mathcal{N}(x_i)} w_{ij} y_j\|^2 \end{aligned} \quad (5.6)$$

It is easy to see that y is free of translation in Eq. 5.6. For every point y , if both y and its neighbors are translated for the same distance, the RHS of Eq. 5.6 keeps unchanged. As a result, there will be many trivial solutions to minimize Eq. 5.6. In order to remove this freedom, zero-mean constraint is imposed.

$$\sum_i y_i = 0 \quad (5.7)$$

A similar reason as above, if both y and its neighbors are rotated by the same angle, minimizing Eq. 5.6 will also have many trivial solutions. The unity variance constraint is imposed to remove this freedom in solving the above equation.

$$\frac{1}{n} \sum_i y_i y_i' = 1 \quad (5.8)$$

In step 2, the obtained individual weights can be reorganized as an $n \times n$ matrix W , where $W_{ij} = 0$ if j -th data point is not a neighbor of i -th data point,

and $W_{ij} = w_j$ for the j -th neighbor of the i -th data point. From this, we can obtain a matrix M , such that

$$M_{ij} = I - W_{ij} - W_{ji} + \sum_k W_{ki} W_{kj} \quad (5.9)$$

And Eq. 5.6 can be denoted as a matrix form as

$$S_2 = \sum_i \sum_j M_{ij} (y_i \cdot y_j) \quad (5.10)$$

With the constraints stated in Eq. 5.7 and 5.8, a straightforward way to minimization Eq. 5.6 (or Eq. 5.10) is to find the bottom some eigenvectors of the matrix M

Although both step 2 and step 3 are solving least square problems, there are fundamental differences between them. In step 2, we can decompose Eq. 5.1 to n separate linear systems of $K \times K$ equations. In other words, we are trying to abstract the intrinsic characteristic of every single neighborhood alone. Also noted that if $K < p$, where p is the dimension of the data, we need to regularize these equations by adding a small positive value to all the diagonal elements in the matrix. However, in step 3, Eq. 5.6 is not decomposable, we need to solve n points as a whole. The implication is that all the neighborhoods interact at this step, which will confine the global structure of the manifold implicitly.

5.4 Experiments

One of the very important goal of ISOMAP and LLE is to reveal the global structure (normally nonlinear) for a data manifold and to determine the possible

Figure 5.1: Derived one dimensional embedding from a 3D spiral shown in Fig. 1.2 using PCA or metric MDS using Euclidean distances.

underlying factors (latent variables) which govern the global structure. In this section, first, we will try to "decrypt" the structure shown in Fig. 1.2. Then, some experiments are conducted based on some objects selected from the COIL-100 [70] database from Columbia University.

5.4.1 Experiment I: Spiral Data

As pointed out above, the manifold described in Fig. 1.2 is actually governed by a single variable $t \in [0, 1]$, such that $\mathbf{M} = \{\mathbf{f}(t) = [\sin 2\pi t, \cos 2\pi t, t]'\}$. Although the global structure is in 3-dimension, the intrinsic dimension is only one. To derive the only dimension, PCA and MDS will always fail, because there is no single universal metric to govern the whole data spread. On the contrary, ISOMAP and LLE are designed to deal with such cases. The result from PCA or metric MDS with Euclidean distances is depicted in Fig. 5.1, and the results from both methods are presented in Fig. 5.2.

In Fig. 1.2, all three axes are real numbers, so every point comprises three real values. The index of sample is solely represented through the color. The sample 1 to 1000 is colored from red to yellow, then green, then cyan, then blue,

Figure 5.2: Derived one dimensional embedding from a 3D spiral shown in Fig. 1.2 using ISOMAP and LLE.

then purple, finally back to red. However, in both Fig. 5.1 and Fig. 5.2, only the x-axis shows the embedded value. The y-axis is the sample index, together with the color scheme, shows the index of samples.

It is clear that both ISOMAP and LLE perform perfectly on "decrypt" the underlying nonlinear structure of the spiral $\mathbf{M} = \{f(t) = [\sin 2\pi t, \cos 2\pi t, t]\}$. Both diagrams (a) and (b) in Fig. 5.2 clearly indicate that all the data points indeed lie on a linear one dimensional line. However, Fig. 5.1 fails to find the linear structure of the segment of the spiral.

5.4.2 Experiment II: Two Objects from COIL-100

The experiments are conducted based on the Columbia Object Image Library (COIL-100) [70]. The COIL-100 is a color image database of 100 objects, and all images are with black background. Every object is rotated by a motor in step of 5 degrees, and an image is taken at each step. As a result, every object comprises 72 images with different poses.

I select two objects, object 14, a cat toy, and object 74, a duck toy in

the experiments. Before doing any experiments, all the images were converted to gray scale, and resized from 128×128 to 16×16 in order to reduce the computation load.

First, the embeddings derived from both ISOMAP and LLE for both the cat toy and the duck toy are presented in Fig. 5.3. During the experiments, the neighborhood factor k is set to 6. All the legend is the same as described above for Fig. 5.2.

Two clear observations can be made. First, for each object, ISOMAP and LLE find similar structures. Second, two objects possess very different structures.

The first point needs no more explanation, because both ISOMAP and LLE can learn the global structure for a data manifold perfectly in this case. The reason why the derived structures for toy cat and toy duck are so different can be explained this way. According to the nature of the COIL-100 database, there is only one underlying factor (latent variable), the pose, which varies from 0^0 to 360^0 , that is why we can project all the 72 images to a one-dimensional continuous line. However, since the shape of the two objects are very different, so the appearance to the pose is also very different. Fig. 5.4 and Fig. 5.5 listed 36 images for each object, and the pose interval is 10 degrees, which means that every other images are listed here.

From Fig. 5.3, we can see that the duck toy object only shows one symmetric curve with a peak point in the middle, i.e. 180^0 . However, the cat toy objects give one continuous curve but with two symmetric portions, which means that middle portion, 0.25 to 0.65 of the total samples (i.e. about 90^0 to 240^0), are quite similar. Take a look at Fig. 5.4 and 5.5, we can clearly find that this

observation reflects the truth of the data spread.

5.5 Discussions

The original motivation for both ISOMAP and LLE are from the Psychological research, and [71][20][8] are all initiated from the perception and psychology related topics. Unsurprisingly, in machine learning and pattern recognition, these two also stimulate immense interests and are proved to be useful in some cases. In this section, I discuss the Pros and Cons first. Following that, the connections to the framework of this thesis are discussed. Lastly, some possible future directions are discussed.

5.5.1 Pros and Cons of ISOMAP and LLE

Curse of dimensionality [4] has been known by researchers and engineers for a long time. Many dimensionality reduction techniques have been introduced in literature. Unfortunately, most of them are derived based on linear assumptions. However, in practise, we do meet some cases that the data spread is nonlinear globally but locally they are linear. In 1997, Kambhatla and Leen [72] proposed an approach, Vector Quantization PCA Algorithm (VQPCA), to address such problems in a hard way. VQPCA produces some hard partitions as locals subject to certain criteria. Then it performs the PCA computations on each local. Finally, a set of small local subspaces are obtained. As a result, although VQPCA can reveal each local structure, it is not able to describe the global structure.

ISOMAP and LLE consider the local and global structures differently; how-

ever, the local and global structures are linked. ISOMAP explicitly applies different distance metrics to neighboring and non-neighboring sample points, but during embedding the distances are treated the same. As a result, we obtain a unified global nonlinear structure with local linearity. Contrary to the ISOMAP, LLE does not concern about the global structure explicitly, but through the interaction of all locals, it targets preserving the global properties.

For LLE, to minimize the cost function S_1 (Eq. 5.1) is solving some linear equations, and to minimize the cost function S_2 (Eq. 5.6) can be considered as a simple eigen problem. LLE is attractively simple in terms of implementation.

Besides all the advantages we gained, ISOMAP and LLE share some disadvantages. First, both methods are sensitive to the data density, if the local density is too low, we cannot form a neighborhood over the region. Second, there is no dimensionality reduction mapping defined. The way to gain a mapping is through a simple computation on the $[X, Y]$ pairs, which tends towards local minima. Third, there is no fool-proof way to determine the neighborhood. For k -neighborhood, k is difficult to determine. Locality implies a small neighborhood, but not so small that no neighbor is available. The same effect goes to r -neighborhood.

5.5.2 Connection to The Framework of This Thesis

Although both ISOMAP and LLE make a target of finding the global nonlinear structures of some data spreads, and both of them "Think globally", LLE just "Fit locally". In contrast, ISOMAP always sticks to the "Global structure". In other words, the standing point of ISOMAP is to map the global structure of a manifold to a lower dimensional space, and the principal idea of LLE is to

preserve all the local linearities.

From the above, ISOMAP can naturally fit into the framework of this thesis, but LLE cannot be directly interpreted under the framework. The reason is that ISOMAP can get a single squared distance matrix D^2 , although the neighboring and non-neighboring points have different distance metrics. On the other hand, it is hard to get a single squared distance matrix for LLE, because during the second optimization, the distances keep changing to adapt to the optimization.

5.5.3 Future Directions

Currently, the main application of ISOMAP and LLE is to derive the underlying factor for an object, which probably is a nonlinear manifold. Just as what we have seen in the toy object experiments, the underlying factor has really been revealed such that some very high dimensional images of an object can be embedded into a one-dimensional line. However, due to the differences between objects, the two embedded manifolds are different, as I have commented above, in this case, it is because the appearance of the two objects are different.

Intuitively, we can differentiate two objects based on their distinctive learned manifolds. Fig. 5.6 (a) plotted two manifolds for cat and duck objects derived from ISOMAP, and the two manifolds in (b) is derived from LLE.

It is interesting and exciting to see that whether we can find a function perfectly describing each manifold, and then we can differentiate objects according to the manifold it lies in or nearby.

Figure 5.3: Two objects are embedded using ISOMAP and LLE with the neighborhood factor $k=6$

Figure 5.4: 36 cat toys, and the pose interval is 10 degrees.

Figure 5.5: 36 duck toys, and the pose interval is 10 degrees.

Figure 5.6: One-dimensional manifolds for both the cat and duck objects are plotted in one diagram.

Chapter 6

Embedding Techniques for Face Recognition

6.1 Review of Face Recognition

Face recognition is defined as the biometric identification by scanning a person's face and matching it against a library of known faces. Automatic human face recognition has generated great interests and immense research activities in the past few decades; for detailed surveys see [73] [74] [75]. Face is a useful and convenient biometric characteristic, and recognizing a face automatically by a machine surely will play an important role in future. For example, it will be wonderful that all video surveillance systems can recognize faces correctly, so that automatic security access control can be achieved. On the other hand, face recognition is not an easy task. When the pictures of a face are captured with different lighting conditions or from different angles, they might appear to be very different. When a person is in different moods, the pictures taken are

different,. Not to mention that people's faces keep changing from time to time, growing nice mustache for a gentleman or wearing heavy makeup for a lady will change the appearance drastically as well.

In the 1970s, Kanade [76] proposed a face recognition approach that based on facial geometric features. This type of system performs recognition based on facial components, such as eyes, nose and mouth extracted from the picture. Since only some components are extracted and related geometric information is retained, the representation can be compact. However, till now, there is still no reliable automatic component extraction approach. Moreover, the performances of those geometric feature based approaches are disappointing most of time [77].

In the early 1980s, template matching approach [78] was proposed. Template matching approach uses a set of multiple view instances to represent a face. And for each view instance, it takes the raw intensities as inputs. Template matching approach performs better than the geometric feature based approach in general [79]. However, since all the intensity pixels are taken as inputs, the processing system suffers from the heavy computations.

Since the 1990s, subspace-based approaches have demonstrated excellent performance, and to some extent, dominated the approaches in face recognition [80]. Subspace method also takes the raw intensity pixels as inputs; however before any recognition process, a dimensionality reduction is performed.

Those subspace-based approaches are mostly originated from Turk and Pentland's eigenface [46] that is motivated by the ground-breaking work of Kirby and Sirovich [81], where the authors used the principal components to represent face images. The eigenface is based on the principal component analysis (PCA) [6]. Motivated by the success of the eigenface, some further works are also reported

in literature. Pentland, Moghaddam et al. proposed view-based eigenspaces [82] and probabilistic subspace learning [83], which then led to Bayesian Face Recognition [80]. Since PCA preserves the Euclidean distance between any two instances as discussed in Chapter 3, eigenface is a modified template matching approach, where the representation is getting compact by PCA to alleviate the computation required in recognition stage. Another important subspace-based approach is the fisherface [84] [85] [86] based on fisher's linear discriminant analysis (LDA) [2]. LDA maximizes the separability between different classes, so the dimensionality reduction can also be interpreted as a feature selection.

Both the eigenface and the fisherface transform a given set of face images into a smaller set of basis images using matrix decomposition techniques. Given a new face image, it is projected into a reduced dimensional space spanned by these basis images, where the recognition is carried out by comparing the distances between this new face to all those known ones. To some extent, both approaches have become common performance benchmarks in face recognition.

Since the introduction of the Kernel PCA [14] and Kernel LDA [63], these two techniques were naturally ported to face recognition to enable nonlinear face feature extraction. Yang et al. [87] reported their kernel eigenface in 2000. Following that, in 2002, Yang [88] [89] reported the kernel fisherface as well.

6.2 Validity of Embedding Techniques for Face Recognition

In face recognition, we are given a training set $\mathcal{D} \subseteq \mathbf{X} \times \mathcal{T}$ of n instances $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \dots, (\mathbf{x}_n, t_n)\}$, where $\mathbf{X}$ denotes the original face space and the

set of all g possible persons are labelled by $\mathcal{T}$. Here, we represent the data by an $n \times v$ matrix X that denotes n instances of v (v equals to the length of the feature size) feature variables each. The label is represented by a $g \times n$ matrix G where $g_{ij} = 1$ if instance j belongs to class i and $g_{ij} = 0$ otherwise.

Both eigenface and fisherface are classical face recognition methods, they reduce the dimension of the face data, then perform recognition in the lower dimensional space. The kernel versions of these two methods also follow the same procedures. In the previous chapters, I have discussed that these methods could all be interpreted in terms of embedding techniques, where all pairwise distance relationships are preserved. Still, no practical applications have been used to illustrate the idea of dimensionality reduction by using embedding techniques.

From the embedding viewpoint, eigenface treats all the instances the same, and calculates all the pairwise distances, and then uses the Metric MDS to embed these distances. On the contrary, fisherface first selects every class mean as the sole representative of each class. Then it takes the class information g in the between-class pairwise distances calculation. Both eigenface and fisherface take the assumption that the pure distances can separate the faces of different persons. Fisherface takes a further step assume that the face images of a single person is more similar than the images from different persons. That is the reason why the class mean alone can be used to train the fisherspace.

In an ideal case, all the face images from a single person form a very tight cluster, and the images from different persons separate far way. If this assumption is satisfied, the embedding process can be modelled in this way. To accommodate the first person, the machine draws an axis x_1 , and the images

from that person is allocated around a point on that axis, i.e., the origin¹. When the second person comes, the machine spreads two persons on the two sides from the origin along x_1 . When the third person comes, it draws another axis x_2 perpendicular with x_1 , so that all the images are allocated on the x_1x_2 plane. When the fourth person comes, a third axis x_3 is drawn to construct a 3-dimensional space. The accommodating of images is based on the distances to all previously known typical representatives of each person. Such an embedding process can be illustrated diagrammatically in Fig. 6.1, where face images from three persons are embedded.

Figure 6.1: An ideal example of allocating three persons into a plane (2-dimensional space), where these three persons are separated clearly, and images from the same person are clustered tightly.

¹Spreading images around origin is to ensure that the class mean will be 0 to avoid multiple alternative locations. For further insertions of other persons, the total mean should be always kept at 0

6.3 Experiments

6.3.1 The Face Database

The experimental data comprise two publicly available face databases, AT&T and Yale face databases². The AT&T database contains 40 persons, and each person with 10 images. The variations of those 400 images are mainly due to the scale and pose of the person in the image. In the Yale database, there are 15 persons with 11 images from each person. The variations of those 165 images are mainly due to the facial expressions and lighting conditions. In order to reduce the computational load, all the images in both databases are down-sampled to 20 x 20 pixels.

In all the experiments, the available databases are partitioned evenly (roughly half and half partition of the data set) for training and testing purposes. For both AT&T and Yale databases, we randomly select 5 images from each person to serve as the training set. In order to make maximum use of the available data for testing and to obtain confidence intervals on the recognition rates, we use a 50-fold Cross-Validation (CV) analysis. Each time, 5 images from each person are randomly selected as training set, and the rest is for testing purpose. Finally, the resulting 50 recognition rates are pooled to compute the mean and standard deviation of the recognition rates.

²Thanks to AT&T Laboratory Cambridge and Center for Computational Vision and Control at Yale University for making these two databases publicly available.

6.3.2 Experimental Results

In this thesis, I have discussed some well-known dimensionality reduction techniques such as PCA, LDA, Kernel PCA, and Kernel LDA. And all these techniques have been widely used in face recognition, amongst them, some can be interpreted in the embedding techniques context. In this section, nine sets of experiments on face recognition will be conducted, and they are,

1. PCA based face recognition (i.e., eigenface).
2. Eigenface's embedded version. In this case, the pairwise Euclidean distances between every two training instances are embedded.
3. LDA based face recognition (i.e., fisherface)
4. Fisherface's embedded version. Three steps are involved in the experiment. Firstly, the within-class scatter matrix S_W is obtained from the training face images. Secondly, all the class means of the training image sets are worked out. Finally, the Mahalanobis distances based on S_W between every two class means are embedded.
5. A simplified version of 4. In this case, all the class means are embedded preserving the pairwise Euclidean distances instead of the Mahalanobis distances based on S_W . Zhang et al. [56] named this special case Proto-face.
6. Kernel PCA based face recognition (i.e., kernel eigenface).
7. Kernel eigenface's embedded version.
8. Kernel LDA based face recognition (i.e., Kernel fisherface).

Table 6.1: Average recognition accuracy for AT&T and Yale face databases

	AT&T			Yale		
	Min	Ave	Max	Min	Ave	Max
Eigenface	82.5%	88.5%	94.0%	68.9%	76.3%	84.4%
Embedded Eigenface	81.5%	88.4%	93.5%	70.0%	74.8%	85.6%
Fisherface	88.5%	92.8%	97.0%	92.2%	96.4%	100.0%
Embedded Fisherface	89.5%	93.4%	97.0%	92.2%	96.5%	100.0%
Protoface	83.5%	90.4%	95.0%	70.0%	80.2%	87.8%
Kernel Eigenface	82.0%	88.0%	90.0%	67.8%	75.7%	84.4%
Embedded K-Eigenface	83.0%	87.1%	91.5%	65.6%	72.2%	81.1%
Kernel Fisherface	92.5%	96.3%	98.5%	85.6%	93.6%	98.9%
Convolution Kernel	83.0%	88.0%	92.5%	85.6%	92.8%	97.8%

9. Face recognition in the convolution kernel embedding generated space.

Figure 6.2: Recognition accuracy of nine methods on two databases. In each panel, from left to right, the nine bars record the eigenface, embedded eigenface, fisherface, embedded fisherface, protoface, kernel eigenface, embedded version of kernel eigenface, kernel fisherface, and convolution kernel embedding, respectively.

The detailed results are reported in Table 6.1, and Fig 6.2 also diagrammatically gives the error bars for each method on both databases.

6.3.3 Observations and Reasoning

All the methods in the experiments belong to the subspace family, and they are working fine on the two databases. Three observations are going to be discussed here.

Fisherface works better than eigenface either in the original space or in the kernel space. This result is expected. PCA is used to extract the features based on the criterion of preserving the variances in the samples. Equivalently, it preserves the Euclidean distances between any two instances. As a result, it does not gain any discriminant power if the classification is based on the Euclidean distance in the subspace. LDA, on the other hand, maximizes the ratio of between-class variances to the within-class variances, through this optimization process, it gains better separability than the PCA. In terms of the distance context, it embeds all the class means based on the Mahalanobis distances, where the influences from the variances in the same class is eliminated, such that the discriminant information is gained. Swets and Weng [84] described that the eigenface only derived the Most Expressive Features (MEF), while the LDA derived the Most Discriminant Features (MDF). Although the arguments above are based on the original space, the same arguments are applicable to the kernel space too.

Surprisingly, there is no much improvement, if not even worse, in terms of the recognition accuracy in the kernel space than in the original space. In Yale database, the fisherface performs the best. From Fig. 6.2, we can see that although in most cases the kernel version performs a little bit better than its counterpart in the original space, the gain is not very clear. Even worse, in Yale database, the fisherface performs better

than the kernel fisherface. In all the experiments, the Gaussian RBF function, $k(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{\delta})$, is chosen to be the kernel function, where the constant factor δ is important to the final results, meanwhile, it is also very hard to choose. One reason of the phenomenon may due to the selection of the constant δ . The Gaussian RBF is a bell shaped function, i.e, it is centered around a sample x_i , where it is maximum, and vanishes as the Euclidean distance between x_j and x_i increases. The constant δ controls the speed of the decay; however, there is no theoretical way to derive a perfect δ ; only empirical means exist to determine the value.

The two databases, AT&T and Yale, show very different performances under the same method. Either PCA or LDA takes its assumptions. Although PCA itself does not require multivariate normal assumption, there are quite a number of approaches which assume that the samples are i.i.d. samples from one Gaussian distribution, probably for purpose of graphical interpretation[53] [90]. LDA takes the class information into account, and assumes that all the within-class variance matrices $S_W(i)$'s are the same. Different data sets have their own characteristics, which are not supposed to be the same. As a result, the same method may perform differently on different data sets. From the experiments, we can see that the AT&T database possesses better discriminant information than the Yale database in the original data form.

6.4 Discussions

This chapter discussed some special face recognition methods, namely subspace methods for face recognition, and their embedding interpretations. During the

training phase, some embedding techniques select the class means first, and then only embed those class means instead of all the training samples. So it is questionable that the class means are adequate to represent all the classes, and the obtained space can accommodate all the samples.

Note that the LDA based fisherface indeed selects the class means as the representatives of their corresponding classes. The big success of the fisherface may also suggest that the class means are adequate representatives ³. However, whether the class means spanned space is enough to accommodate all the training samples is still a big question. In the following section, I will discuss these questions.

6.4.1 Theoretical Analysis

Three important scatter matrices were discussed again and again in almost all the data mining and pattern analysis textbooks. They are the total scatter matrix, between-class scatter matrix and within-class scatter matrix. Classical PCA based method spans a subspace based on the first few eigenvectors of the total scatter matrix, just like what has been done in eigenface, which is considered as the one conserves the variance most. In the embedding techniques, if only the class means are embedded, by means of duality of analysis of variance and analysis of distance [91], we are actually eigendecomposing the between-class scatter matrix, and use the first few eigenvectors of the between-class scatter matrix to span a subspace. In order to answer the question raised above, we are mainly interested in the fact that if we are to take eigenvectors

LDA is also subject to the constraint $\omega^T S_W \omega = 1$. However, as shown in section 3.4.2.1, this constraint just enforces a whitening preprocess, and then LDA embeds the classes means. Equivalently, LDA embeds all the class means based on the pairwise Mahalanobis distances between any two class means.

corresponding to large eigenvalues and span a space using those eigenvectors, the between-class scatter matrix should be focused on.

Given a set of labelled data $\{(\mathbf{x}_i, t_i)\}$, $i \in \{1, 2, \dots, n\}$, $\mathbf{x}_i \in \mathbb{R}^v$, $t_i \in \{1, 2, \dots, g\}$. Let data matrix $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)'$ an $n \times v$ matrix and the class indicator matrix G with size $g \times n$, in which $G(i, j)$ will be assigned 1 if the j th point belongs to the i th class, otherwise 0 will be assigned instead. Let $\mathbf{g}_j$ denote the j th row of G , so each class has $n_j = \mathbf{g}_j \mathbf{1}$ samples and the orthogonal projection along the mean of class j is $P_j = \frac{1}{n_j} \mathbf{g}_j' \mathbf{g}_j$. For the whole data set, the total projection matrix will be $P = \frac{1}{n} \mathbf{1} \mathbf{1}'$. The overall mean and the class means are

$$\mu = \frac{1}{n} \mathbf{1}' X; \quad \mu_j = \frac{1}{n_j} \mathbf{g}_j X \quad (6.1)$$

Let S_T , S_B and S_W denote the total scatter matrix, between-class scatter matrix and within-class scatter matrix respectively.

$$\begin{aligned} nS_T &= \sum_{i=1}^n (\mathbf{x}_i - \mu)' (\mathbf{x}_i - \mu) = X'(I - P)X \\ &= X'Q_TX = (XQ_T)'(XQ_T) \end{aligned} \quad (6.2)$$

$$\begin{aligned} nS_B &= \sum_{i=1}^g n_i (\mu_i - \mu)' (\mu_i - \mu) = X' \left(\sum_{j=1}^g P_j - P \right) X \\ &= X'Q_BX = (XQ_B)'(XQ_B) \end{aligned} \quad (6.3)$$

$$\begin{aligned} nS_W &= n(S_T - S_B) = X' \left(I - \sum_{j=1}^g P_j \right) X \\ &= X'Q_WX = (XQ_W)'(XQ_W) \end{aligned} \quad (6.4)$$

For any matrix U , if $U = U^2$, it is an orthogonal projection matrix. It is clear that Q_T , Q_B and Q_W are appropriate orthogonal projections. Also note that $Q_T = Q_B + Q_W$, $Q_T Q_W = Q_W$, $Q_T Q_B = Q_B$, $Q_B Q_W = NULL$.

Consider the following generalized eigenvalue problem

$$Q_T \nu = \lambda Q_W \nu \quad (6.5)$$

It is clear that the generalized eigenspaces are Q_B , Q_W and P , and an interesting point is investigated here,

- Any vector $\nu \in Q_B \mathcal{R}$ (or ν is a vector projected by Q_B) is a generalized eigenvector with eigenvalue $\lambda = \infty$.

Proof: RHS of Eq. 6.5 equals to $Q_T Q_B$, which equals to Q_B . LHS of Eq. 6.5 equals to $\lambda Q_W Q_B$, which equals to $X0$, in order to make Eq. 6.5 to be true. λ shall be as large as possible, or ∞ .

As a conclusion, if eigenspaces corresponding to large eigenvalues are interested, Q_B shall be focused on. In almost all eigenspace methods, only the first few components corresponding to large eigenvalues are entertained. As a sequel, the between-class scatter matrix is more useful in constructing a eigendecomposition-based subspace. Moreover, we are not interested in the optimal reconstruction in most classification problems, and a subspace which can accommodate all the sample points is enough.

Finally, a toy example is given to show the correctness of the above arguments. There are three classes of data. Class one has two data points $[10, 8, 6, 4]$ and $[9, 7, 7, 3]$. Class two has three data point $[5, 2, 1, -1]$, $[4, 3, 0, 0]$ and $[6, 3, 1, 0]$. Class three has three data points $[0, 1, -5, -4]$, $[-1, 0, -7, -3]$

$$Q_B = \begin{bmatrix} 0.375 & 0.375 & -0.125 & -0.125 & -0.125 & -0.125 & -0.125 & -0.125 \\ 0.375 & 0.375 & -0.125 & -0.125 & -0.125 & -0.125 & -0.125 & -0.125 \\ -0.125 & -0.125 & 0.2083 & 0.2083 & 0.2083 & -0.125 & -0.125 & -0.125 \\ -0.125 & -0.125 & 0.2083 & 0.2083 & 0.2083 & -0.125 & -0.125 & -0.125 \\ -0.125 & -0.125 & 0.2083 & 0.2083 & 0.2083 & -0.125 & -0.125 & -0.125 \\ -0.125 & -0.125 & -0.125 & -0.125 & -0.125 & 0.2083 & 0.2083 & 0.2083 \\ -0.125 & -0.125 & -0.125 & -0.125 & -0.125 & 0.2083 & 0.2083 & 0.2083 \\ -0.125 & -0.125 & -0.125 & -0.125 & -0.125 & 0.2083 & 0.2083 & 0.2083 \end{bmatrix}$$

$$Q_W = \begin{bmatrix} 0.5 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.6667 & -0.3333 & -0.3333 & 0 & 0 & 0 \\ 0 & 0 & -0.3333 & 0.6667 & -0.3333 & 0 & 0 & 0 \\ 0 & 0 & -0.3333 & -0.3333 & 0.6667 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.6667 & -0.3333 & -0.3333 \\ 0 & 0 & 0 & 0 & 0 & -0.3333 & 0.6667 & -0.3333 \\ 0 & 0 & 0 & 0 & 0 & -0.3333 & -0.3333 & 0.6667 \end{bmatrix}$$

Through a simple matrices multiplication, it is clear that $nS_T = X'S_TX$ is correct.

$$X'Q_TX = \begin{bmatrix} 115.5 & 77.75 & 147.75 & 82.25 \\ 77.75 & 58.875 & 100.375 & 58.625 \\ 147.75 & 100.375 & 195.875 & 107.125 \\ 82.25 & 58.625 & 107.125 & 63.875 \end{bmatrix}$$

And, Q_TQ_W is exactly the Q_W , Q_BQ_W is a NULL.

6.4.2 Empirical Analysis

Again, take the AT&T face database as an experimental database. And an empirical approach can help us to find out an interesting phenomenon. Let the total scatter matrix and between-class scatter matrix spanned subspaces work as two distributed classes, and we want to find out whether these two classes are with linearly separated or embedded distributions. One simple approach of visualizing the fact is to plot their mutual principal components, just as what has been done in [80]. The details are to get the principal components from both the total scatter matrix and between-class scatter matrix, then a 3D scatter plot of first three components from each class can be put in a single plot. Fig 6.3 illustrates a plot for the AT&T database.

Figure 6.3: Two hypothetical distributions denote two classes, one is for the total scatter, and the other is for the between-class scatter. The first three principal components from these two distributions are shown in a single 3-dimensional scatter plot. The circles for between-class scatter and the crosses for total scatter.

It shows complete enmeshed distributions in the Fig 6.3, and this phenomenon intuitively tells that the between-class distribution has larger cov-

erage. In other words, the first few components of between-class scatter matrix spanned space will have a bigger coverage than the space spanned by the first few components of the total scatter matrix. Or, if the total scatter matrix spanned subspace can house all the face images, so can the between-class scatter matrix spanned space.

Chapter 7

Conclusions & Recommendations

7.1 Conclusions

Serving different purposes, dimensionality reduction techniques can be categorized into two. The first category of techniques are designed to reduce the dimensionality so as to mitigate the computational load in the subsequent processing. And the second category of techniques are to find the latent variables for the data spread or manifold. Most category one techniques are variance based techniques, such as PCA and LDA, which can be interpreted as embedding techniques through metric MDS. Meanwhile, recently developed category two techniques, such as ISOMAP and LLE, are closely related to the embedding techniques. This thesis intended to unify these two categories of techniques under one roof, the embedding family of techniques through MDS. Under such a framework, these conventional techniques can be considered either selecting

Table 7.1: Distance metrics used in dimensionality reduction techniques

Name of techniques	Distance metric
PCA	$d_{ij}^2 = (x_i - x_j)(x_i - x_j)'$
LDA	$d_{ij}^2 = (x_i - x_j)S_W(x_i - x_j)'$
Kernel PCA (Gaussian Kernel)	$d_{ij}^2 = 2 - 2 \exp(-\frac{\ x_i - x_j\ ^2}{\delta})$
Convolution Kernel (Direct Sum)	$d_{ij}^2 = \begin{cases} 2 - 2 \exp(-\frac{\ x_i - x_j\ ^2}{\delta}) & \text{if } t_i = t_j \\ 4 - 2 \exp(-\frac{\ x_i - x_j\ ^2}{\delta}) & \text{if } t_i \neq t_j \end{cases}$
ISOMAP	$d_{ij}^2 = \begin{cases} (x_i - x_j)(x_i - x_j)' & \text{if } \textit{neighboring} \\ \text{Dijkstra's algorithm} & \text{if } \textit{nonneighboring} \end{cases}$

different representatives or using different distance metrics to construct the squared distance matrix, or both. Table 7.1 demonstrates the distance metrics used for some embedding version of techniques, where x_i and x_j are two data points; S_W is the within-class scatter matrix; t_i and t_j are the class labels for x_i and x_j , respectively. Among the techniques listed in Table 7.1, LDA only embeds all the class means, while all others embed all the data points.

The dimensionality reduction techniques belonging to the first category play a key role in some data analysis tasks, such as pattern recognition, for it either extracts or selects features for the future process. In order to obtain a suitable description with a shorter length for intended tasks, traditionally, a multivariate analysis model is required. Under such a setting, in literature, there are two commonly-used techniques, PCA and LDA. Both techniques are variance analysis techniques, and certain variance criteria shall be satisfied. For PCA, the criterion is that the lower dimensional representation can preserve most of the variance, which guarantees the best reconstruction. As a result, sometimes the extracted lower dimensional description y is named Most Expressive Feature. LDA, on the other hand, takes the class information in the training to satisfy a criterion that the ratio of the between-class variance and within-class variance

is found to hit the maximum. In a sequel, it gains the best separability, and sometimes named as Most Discriminant Feature.

However, from an embedding or geometric point of view, given a set of training samples $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, the pairwise distances can be obtained easily, and the number of such distances is $\binom{n}{2}$. The dimensionality reduction process is to embed all the training points $\mathbf{x}$ into a p -dimensional space, such that the obtained embedded points $\{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\}$ preserve the pairwise distances from the original ones. Metric MDS can be applied to embed all the samples. Through the analysis, both PCA and LDA can be interpreted as a special metric MDS. The only two differences between embedded PCA and LDA are in that they select different representatives and they choose different distance metrics. PCA takes all training samples but LDA only takes all class means as representatives. PCA uses the Euclidean distances but LDA utilizes the Mahalanobis distances.

Recently, kernel methods are widely accepted in the machine learning and pattern recognition community. The success of such a family of methods lies in a technique called the kernel trick, by which all the training samples are projected into a very high dimensional feature space implicitly. The implicitness means only manipulating the dot products in the original space is required. Since the dot product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ is a kind of similarity measure of two instances $\mathbf{x}_i$ and $\mathbf{x}_j$, and similarity is the inverse form of dissimilarity, there is a possibility to interpret the kernel methods in terms of an embedding technique as well. Interpreted as metric MDS, the only difference between KPCA and PCA is that they embed different distance metrics. PCA embeds the normal Euclidean distance, but KPCA embeds a distance metric which is generated from the

kernel function. Convolutional kernel can generate new kernels from existing kernels, by incorporating this technique as ad-hoc discriminant distance metric is derived based on both the data sample descriptors and the class information. A further embedding process can possibly generate a metric space with gained discriminant information.

The techniques belonging to the first category, such as PCA and LDA, are targeting to a cloud of data. In such cases, the data samples are spread in a homogenous space, or each dimension is treated the same in either describing a data sample or differentiating two data samples. Metric MDS with Euclidean distances are also targeting the same situation. However, in some circumstances, the data spread is a nonlinear manifold, such as a one-dimensional spiral in Fig. 1.2. It is hard to generalize the conventional variance analysis techniques to deal with such data spread; however, the embedding techniques can be extended to meet such needs, which are the second category of techniques in this thesis. Due to the fact that for such data spread the structure is locally linear and globally nonlinear, ISOMAP sets different distance metrics for neighboring points and non-neighboring points explicitly to successfully reconstruct the nonlinear structure of such a data spread. On the other hand, two consecutive optimization steps are utilized to preserve both local and global information for the data spread in LLE. In the first step, the local information is preserved, and the second step tries to preserve the global structure through interactions between every two locals.

Since both PCA and LDA have been utilized in face recognition with the name eigenface and fisherface, face recognition is conducted based on both the category one conventional techniques and their embedding interpretations in

the experimental chapter. In addition, a convolution kernel generated ad-hoc distance metric is applied to both AT&T and Yale databases as well. The objectives of the set of experiments include, first, to evaluate the equivalence between the conventional methods and their corresponding embedded versions. Second, to prove that the framework can be generalized by tuning the distance metrics used. A convolution kernel generated metric is used, but more may be applied in future. Lastly, to show that for both AT&T and Yale databases, the class means are sufficient in training the face space.

7.2 Recommendations for Future Research

Indeed the embedding techniques are example-based methods [92], whose success demands that all the training samples are valid, and possibly informative, representatives for their belonging classes. However, in this thesis, the validity checking and informative measure of the training samples are not very well controlled. Only in section 6.4, I discussed the validity of using class mean (average face) to represent a person for embedding. One invalid sample may severely affected the validity of the obtained space, and less informative samples may generate a small space which is not capacious enough to accommodate all the samples.

As discussed above that the kernel methods are based on all the pairwise similarity measures, it is possible to extend all this family of methods in terms of the embedding techniques. However, in this thesis, I have not derived the sphering (or whitening) process in the kernel space. If the data can be sphered in the feature space successfully based on only the dot products, the Kernel LDA

can immediately be interpreted in terms of an embedding technique. More importantly, the intrinsic connection between the large margin classifier (most notably Support Vector Machine) and conventional linear discriminant classifier will be very clear.

In section 5.5.3, I pointed out that for different objects the embedded curves were different due to the objects' different intrinsic characteristics. In the cat and duck toy example, the difference is that from about 90^0 to 240^0 , the duck toy appears very similar, while the cat toy does not show such characteristics. As such, the two embedded curves are very different. In other words, these curves reflect the specialities of different objects. In future, it will be an interesting direction to find a way to describe the curves, and match an instance according to the "distances" from its projection in the space to all known curves.

Bibliography

- [1] R. O. Duda, P. E. Hart, and D. G. Stork, Pattern Classification, 2nd ed. New York: John Wiley and Sons, 2001.
- [2] K. Fukunaga, Introduction to Statistical Pattern Recognition, 2nd ed. San Diego: Academic Press, 1990.
- [3] C. M. Bishop, Neural Networks for Pattern Recognition. Oxford: Oxford University Press, 1995.
- [4] R. Bellman, Adaptive Control Processes: A Guided Tour. Princeton: Princeton University Press, 1961.
- [5] K. Pearson, "On lines and planes of closest fit to systems of points in space," Philosophical Magazine, Series 6, vol. 2, pp. 559–571, 1901.
- [6] I. T. Jolliffe, Principal Component Analysis. New York: Springer-Verlag, 1986.
- [7] J. B. Tenenbaum, "Mapping a manifold of perceptual observations," Advances in Neural Information Processing Systems 10, 1998.
- [8] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," Science, vol. 290, no. 22, pp. 2323–2326, 2000.
- [9] T. F. Cox and M. A. A. Cox, Multidimensional scaling, 2nd ed. Chapman & Hall/CRC, 2000.
- [10] I. Borg and P. Groenen, Modern Multidimensional Scaling: Theory and Applications, ser. Springer Series in Statistics. Springer, 1997.
- [11] J. C. Gower, "Principal coordinates analysis," in Encyclopedia of Biostatistics, P. Armitage and T. C. J. A., Eds. Wiley, Chichester, 1997, pp. 3514–3518.
- [12] V. Vapnik, The Nature of Statistical Learning Theory. New York: Springer-Verlag, 1995.
- [13] ———, Statistical Learning Theory. New York: John Wiley and Sons, 1998.
- [14] B. Scholkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," Neural Computation, vol. 10, pp. 1299–1319, 1998.
- [15] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," Neural Computation, vol. 12, pp. 2385–2404, 2000.
- [16] B. Scholkopf, "The kernel trick for distances," Advances in Neural Information Processing Systems 13, pp. 1299–1319, 2001.

- [17] C. K. I. Williams, "On a connection between kernel PCA and metric multidimensional scaling," *Advances in NIPS*, vol. 13, 2001.
- [18] Z. Zhang, J. T. Kwok, D.-Y. Yeung, and W. Wang, "A novel distance-based classifier using convolution kernels and Euclidean embeddings," HKUST, Tech. Rep. CS02-28, 2002.
- [19] Z. Zhang, J. T. Kwok, and D.-Y. Yeung, "Parametric distance metric learning with label information," HKUST, Tech. Rep.
- [20] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 22, pp. 2319–2323, 2000.
- [21] A. U. Kennington, *Differential Geometry Reconstructed: A Unified Systematic Framework*. e-book, 2004. [Online]. Available: <http://www.topology.org/tex/conc/dg.html>
- [22] C. K. I. Williams, "On a connection between kernel PCA and metric multidimensional scaling," *Machine Learning*, vol. 46, pp. 11–19, 2002.
- [23] Z. Zhang, "Learning metrics via discriminant kernels and multidimensional scaling: Toward expected euclidean representation," *Proceedings of ICML-2003*, Washington DC, 2003.
- [24] M. J. Greenacre and L. Underhill, "Scaling a data matrix in a low dimensional euclidean space," in *Topics in Applied Multivariate Analysis*, D. Hawkins, Ed. Cambridge University Press, 1982, pp. 183–269.
- [25] K. V. Mardia, J. Kent, and J. Bibby, *Multivariate Analysis*. London: Academic Press, 1979.
- [26] R. Fisher, "The systematic location of genes by means of cross-over ratios," *American Naturalist*, vol. 56, pp. 406–411, 1922.
- [27] J. De Leeuw and W. J. Heiser, "Theory of multidimensional scaling," in *Handbook of Statistics*, P. R. Krishnaiah and L. N. Kanal, Eds. North-Holland, 1982, vol. 2, pp. 285–316.
- [28] G. Crippen, "A novel approach to calculation of conformation: Distance geometry," *Journal of Computational Physics*, vol. 24, pp. 96–107, 1977.
- [29] G. Crippen and T. Havel, *Distance Geometry and Molecular Conformation*. Wiley, 1988.
- [30] R. Shepard, "Stimulus and response generalization: A stochastic model relating generalization to distance in psychological space," *Psychometrika*, vol. 22, pp. 325–345, 1957.
- [31] J. B. Kruskal and M. Wish, *Multidimensional Scaling*. Beverly Hills: Sage Publications, 1978.
- [32] P. Arabie, J. Carroll, and W. DeSarbo, *Three-way Scaling and Clustering*. Sage Publications, 1987.
- [33] J. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, pp. 1–27, 1964.
- [34] J. De Leeuw, "Applications of convex analysis to multidimensional scaling," in *Recent developments in Statistics*, B. J., B. F., R. G., and V. C. B., Eds. North Holland, 1977, pp. 133–145.

- [35] A. R. Webb, "Multidimensional scaling by iterative majorization using radial basis functions," *Pattern Recognition*, vol. 28, no. 5, pp. 753–759, 1995.
- [36] J. de Leeuw, "Block-relaxation algorithms in statistics," *Statistics Seminar, USC*, 1996.
- [37] D. R. Hunter and K. Lange, "Quantile regression via an MM algorithm," *Journal of Computational and Graphical Statistics*, vol. 9, pp. 60–77, 2000.
- [38] K. Lange, D. R. Hunter, and I. Yang, "Optimization transfer using surrogate objective functions (with discussions)," *Journal of Computational and Graphical Statistics*, vol. 9, pp. 1–59, 2000.
- [39] D. R. Hunter and K. Lange, "Computing estimates in the proportional odds model," *Annals of the Institute of Statistical Mathematics*, 2001.
- [40] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Baltimore: The Johns Hopkins University Press, 1996.
- [41] A. K. Jain, R. P. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 1, pp. 4–37, 2000.
- [42] J. E. Jackson, *A User's Guide to Principal Components*. New York: Wiley, 1991.
- [43] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, pp. 498–520, 1933.
- [44] A. Miguel, "Continuous latent variable models for dimensionality reduction and sequential data reconstruction," Ph.D. dissertation, Department of CS, University of Sheffield, UK, 2001.
- [45] B. D. Flury, "Developments in principal component analysis," in *Recent Advances in Descriptive Multivariate Analysis*, W. J. Krzanowski, Ed. Oxford Science Publications, 1995, pp. 14–33.
- [46] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.
- [47] J. C. Gower and W. J. Krzanowski, "Analysis of distance for structured multivariate data and extensions to multivariate analysis of variance," *Journal of the Royal Statistical Society Series C: Applied Statistics*, vol. 48, pp. 505–519, 1999.
- [48] G. Li and J. Zhang, "Sphering and its properties," *Sankhya: The Indian Journal of Statistics, Series A*, vol. 60, pp. 119–133, 1998.
- [49] N. Nguyen, P. Milanfar, and G. Goulb, "Efficient generalized cross-validation with applications to parametric restoration and resolution enhancement," *IEEE Trans. Image Processing*, vol. 10, no. 9, pp. 1299–1308, 2001.
- [50] J. C. Gower, "Adding a point to vector diagrams in multivariate analysis," *Biometrika*, vol. 55, pp. 582–585, 1968.
- [51] C. Cuadras and C. Arenas, "A distance based regression model for prediction with mixed data," *Communications in Statistics, Theory Meth.*, vol. 19, pp. 2261–2279, 1990.
- [52] T. W. Anderson, *An Introduction to Multivariate Statistical Analysis*. New York: Wiley, 1958.

- [53] R. A. Johnson and D. W. Wichern, *Applied multivariate statistical Analysis*, 5th ed. Prentice Hall, 2002.
- [54] J. Gower, "Multivariate analysis and multidimensional geometry," *Statistician*, vol. 17, no. 1, pp. 13–28, 1967.
- [55] J. De Leeuw, "Multidimensional scaling," in *International Encyclopedia of the Social and Behavior Sciences*. Elsevier, 2001.
- [56] Z. Zhang, J. T. Kwok, D.-Y. Yeung, and W. Wang, "A novel family of subspace methods — protofaces and its kernel version," HKUST, Tech. Rep. CS02-29, 2002.
- [57] V. N. Vapnik and A. J. Chervonenkis, "Theory of pattern recognition," Nauka, 1974.
- [58] I. Guyon, B. Boser, and V. Vapnik, "Automatic capacity tuning of very large vc-dimension classifiers," *Advances in NIPS*, vol. 5, 1993.
- [59] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.
- [60] B. Scholkopf, C. Burges, and A. Smola, *Advances in Kernel Methods – Support Vector Learning*. Cambridge: MIT Press, 1999.
- [61] I. J. Schoenberg, "Metric spaces and positive definite functions," *Transactions of the American Mathematical Society*, vol. 44, pp. 522–536, 1938.
- [62] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers," *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, 1992.
- [63] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Müller, "Fisher discriminant analysis with kernels," in *Neural Networks for Signal Processing IX*, Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, Eds. IEEE, 1999, pp. 41–48.
- [64] D. Haussler, "Convolution kernels on discrete structures," 1999, UCSC-CRL-99-10.
- [65] B. Scholkopf, *Learning with Kernels*. The MIT Press, 2002.
- [66] C. L. Blake and C. J. Merz, "UCI repository of machine learning databases," 1998. [Online]. Available: <http://www.ics.uci.edu/~mllearn/MLRepository.html>
- [67] B. S. Everitt, *An Introduction to Latent Variable Models*, ser. Monographs on Statistics and Applied Probability. London: Chapman & Hall, 1984.
- [68] C. M. Bishop, M. Svensén, and C. K. I. Williams, "EM optimization of latent-variable density models," *Advances in Neural Information Processing Systems* 8, 1996.
- [69] L. K. Saul and S. T. Roweis, "Think globally, fit locally: Unsupervised learning of nonlinear manifolds," Department of Computer and Information Science, University of Pennsylvania, Tech. Rep. MS CIS-02-18, 2002.
- [70] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia object image library (coil-100)," Department of Computer Science, Columbia University, Tech. Rep. CUCS-006-96, 1996.
- [71] H. S. Seung and D. D. Lee, "The manifold ways of perception," *Science*, vol. 290, pp. 2268–2269, December 2000.
- [72] N. Kambhatla and T. K. Leen, "Dimension reduction by local principal component analysis," *Neural Computation*, vol. 9, no. 7, pp. 1493–1516, 1997.

- [73] R. Chellappa, C. Wilson, and S. Sirohey, "Human and machine recognition of faces: A survey," *Proceedings of IEEE*, vol. 83, no. 5, 1995.
- [74] T. Fromherz, "Face recognition: A summary of 1995-1997," International Computer Science Institute, Berkeley, Tech. Rep. ICSI TR-98-027, 1998.
- [75] W. Zhao, R. Chellappa, A. Rosenfeld, and P. Phillips, "Face recognition: A literature survey," UMD CfAR, Tech. Rep. CAR-TR-948, 2000.
- [76] T. Kanade, "Picture processing system by computer complex and recognition of human faces," Ph.D. dissertation, Department of Information Science, Kyoto University, 1973.
- [77] R. Brunelli and T. Poggio, "Face recognition through geometrical features," *Proceedings of ECCV*, pp. 792–800, 1992.
- [78] R. J. Baron, "Mechanisms of human facial recognition," *International Journal of Man-Machine Studies*, vol. 15, no. 2, pp. 137–178, 1981.
- [79] R. Brunelli and T. Poggio, "Face recognition: Features versus templates," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, no. 10, pp. 1042–1052, 1993.
- [80] B. Moghaddam, T. Jebara, and A. Pentland, "Bayesian face recognition," *Pattern Recognition*, vol. 33, no. 11, pp. 1771–1782, 2000.
- [81] M. Kirby and L. Sirovich, "Application of the Karhunen-Loève procedure for the characterization of human faces," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 12(1), 1990.
- [82] A. Pentland, B. Moghaddam, and T. Starner, "View-based and modular eigenspaces for face recognition," *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition 1994*, 1994.
- [83] B. Moghaddam. and A. Pentland, "Probabilistic visual learning for object representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 7, pp. 696–710, 1997.
- [84] D. Swets and J. Weng, "Using discriminant eigenfeatures for image retrieval," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, no. 8, pp. 831–836, 1996.
- [85] P. Belhumeur, J. Hespanha, and D. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 7, pp. 711–720, 1997.
- [86] W. Zhao, R. Chellappa, and A. Krishnaswamy, "Discriminant analysis of principal components for face recognition," In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pp. 336–341, 1998.
- [87] M. H. Yang, N. Ahuja, and D. Kriegman, "Face recognition using kernel eigenfaces," *Proceedings of the 2000 IEEE International Conference on Image Processing (ICIP 2000)*, vol. 1, pp. 37–40, 2000.
- [88] M. H. Yang, "Kernel eigenfaces vs. kernel fisherfaces: Face recognition using kernel methods," *Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, 2002.
- [89] C. Andrieu, P. M. Djurić, and A. Doucet, "Model selection by MCMC computation," *Signal Processing*, vol. 81, pp. 19–37, 2001.

-
- [90] T. W. Anderson, *An Introduction to Multivariate Statistical Analysis*, 2nd ed. New York: Wiley, 1984.
 - [91] J. C. Gower, "Some distance properties of latent root and vector methods used in multivariate analysis," *Biometrika*, vol. 53, no. 3/4, pp. 325–338, Dec. 1966.
 - [92] I. Guyon and D. G. Stok, "Linear discriminant and support vector classifier," in *Advances in Large Margin Classifiers*, A. J. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, Eds. Cambridge, MA: MIT Press, 2000, pp. 147–170.