THE UNIVERSITY OF CHICAGO

LOCALITY PRESERVING PROJECTIONS

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
XIAOFEI HE

CHICAGO, ILLINOIS
DECEMBER 2005

ABSTRACT

Many problems in information processing involve some form of dimensionality reduction. In this thesis, we introduce Locality Preserving Projections (LPP). These are linear projective maps that arise by solving a variational problem that optimally preserves the neighborhood structure of the data set. LPP should be seen as an alternative to Principal Component Analysis (PCA) – a classical linear technique that projects the data along the directions of maximal variance. When the high dimensional data lies on a low dimensional manifold embedded in the ambient space, the Locality Preserving Projections are obtained by finding the optimal linear approximations to the eigenfunctions of the Laplace Beltrami operator on the manifold. As a result, LPP shares many of the data representation properties of nonlinear techniques such as Laplacian Eigenmaps or Locally Linear Embedding. Yet LPP is linear and more crucially is defined everywhere in ambient space rather than just on the training data points. Theoretical analysis shows that PCA, LPP, and Linear Discriminant Analysis (LDA) can be obtained from different graph models. Central to this is a graph structure that is inferred on the data points. LPP finds a projection that respects this graph structure. We have applied our algorithms to several real world applications, e.g. face analysis and document representation.

ACKNOWLEDGEMENTS

I would like to thank my advisor Partha Niyogi for his help and thoughtful advice. I am very grateful to Gina-Anne Levow and Pedro Felzenszwalb for their supports. I am especially grateful to Hong-Jiang Zhang and Wei-Ying Ma for motivating me to think about problems in computer vision and information retrieval.

I would like to thank my fellow students Zhimin Xie, Dinoj Surendran, Vikas Sindhwani and Irina Matveeva for conversations. I am especially grateful to Deng Cai for many discussions and insightful suggestions.

TABLE OF CONTENTS

ABSTRACT ii

ACKNOWLEDGEMENTS iii

LIST OF FIGURES vii

LIST OF TABLES x

1 INTRODUCTION 1

2 LOCALITY PRESERVING PROJECTIONS 3
- 2.1 Linear Techniques for Dimensionality Reduction 3
- 2.2 The Algorithm 4
- 2.3 Theoretical Justifications 6
  - 2.3.1 Optimal Linear Embedding 6
  - 2.3.2 Geometrical Justification 8
- 2.4 Connections to Principal Component Analysis 11
- 2.5 Connections to Linear Discriminant Analysis 11
- 2.6 Computational Issues 14
- 2.7 Model Selection for LPP 15
- 2.8 Complexity Analysis 16
- 2.9 Examples 18
  - 2.9.1 Simple synthetic Examples 18
  - 2.9.2 2-D Data Visualization 20
  - 2.9.3 Manifold of Handwritten Digits 24
  - 2.9.4 20 Newsgroups Classification 25

3 KERNEL LPP 29
- 3.1 Derivation of LPP in Reproducing Kernel Hilbert Space 29
- 3.2 Derivation of Locally Linear Embedding Algorithm in Reproducing Kernel Hilbert Space 31

4 FACE ANALYSIS USING LAPLACIANFACES 35
- 4.1 Appearance Based Face Analysis 35
- 4.2 Manifold Ways for Visual Analysis 37
  - 4.2.1 Manifold Learning via Dimensionality Reduction 38
  - 4.2.2 Learning Laplacianfaces for Representation 38
  - 4.2.3 Face Manifold Analysis 40
- 4.3 Experimental Results 44
  - 4.3.1 Face Representation Using Laplacianfaces 44
  - 4.3.2 Face Recognition Using Laplacianfaces 45

4.3.3 Face Clustering Using Laplacianfaces 53
4.3.4 Discussion 54

5 LOCALITY PRESERVING INDEXING FOR DOCUMENT REPRESENTATION 56
5.1 Document Representation 56
5.2 Locality Preserving Indexing 58
5.3 Theoretical Analysis 59
5.3.1 Singular Value Decomposition 59
5.3.2 Geometry of the Document Space: Global VS. Local 60
5.3.3 Connections to Latent Semantic Indexing 61
5.3.4 Connections to Laplacian Eigenmap 63
5.4 Experimental Results 65
5.4.1 Data Preparation 65
5.4.2 Evaluation on Similarity Measure 65
5.4.3 Evaluation on Discriminating Power 72
5.4.4 2D Visualization of the Document Set 79
5.4.5 Document Clustering 79

6 FEATURE SELECTION USING LAPLACIAN SCORE 87
6.1 Feature Selection VS. Feature Extraction 88
6.2 Laplacian Score 90
6.2.1 The Algorithm 90
6.3 Theoretical Analysis 91
6.3.1 Objective Function 91
6.3.2 Connections to Fisher Criterion Score 93
6.4 Experimental Results 96
6.4.1 UCI Iris Data 96
6.4.2 Face Clustering on PIE 98

7 FROM VECTOR TO TENSOR — TENSOR SUBSPACE ANALYSIS 106
7.1 The Algebra of Tensor 107
7.2 Tensor Subspace Analysis 109
7.2.1 Laplacian based Dimensionality Reduction 109
7.2.2 The Linear Dimensionality Reduction Problem in Tensor Space 110
7.2.3 Coordinate System Transformation 110
7.2.4 Optimal Linear Embedding 111
7.2.5 Derivation 113
7.2.6 Computation 115
7.3 Face Recognition with Tensor Representation 116
7.3.1 Experiments on PIE Database 117
7.3.2 Experiments on ORL Database 119

7.4 Image Clustering with Tensor Representation 121
7.4.1 Data Preparation 122
7.4.2 2-D Visualization of Image Set 122
7.4.3 Clustering Results 124

8 CONCLUSIONS 128

REFERENCES 129

LIST OF FIGURES

2.1 The upper plot shows the results of PCA. The lower plot shows the results of LPP. The straight line segments describe the two bases. The first basis is shown as a longer line segment, and the second basis is shown as a shorter line segment. Clearly, LPP is insensitive to the outlier at the upper right corner. 19

2.2 The upper plot shows the results of PCA. The lower plot shows the results of LPP. The straight line segments describe the two bases. The first basis is shown as a longer line segment, and the second basis is shown as a shorter line segment. In this example, LPP has more discriminating power than PCA. 21

2.3 The first plot shows the data set which is composed of three circles. The second plot shows the 2-D projection by PCA. The third plot shows the 2-D projection by LPP. In this example, LPP is capable of discovering the intrinsic topological structure of the data set while PCA fails. 22

2.4 The handwritten digits ("0"-"9") are mapped into a 2-dimensional space. The left panel in the first row is a representation of the set of all images of digits using the Laplacian eigenmap. The right panel in the first row shows the results of LPP using the first two directions to represent the data. The panel in the second row shows the results of principal component analysis. Each color corresponds to a digit. . . . 23

2.5 A two-dimensional representation of the set of images of handwritten digits using the Locality Preserving Projections. As can be seen, the first dimension (horizontal axis) appears to describe the line thickness of each digit, and the second dimension (vertical axis) appears to describe the slant of each digit. 24

2.6 The distributions of 10 testing images in the 2-D plane. As can be seen, these testing samples optimally find their coordinates which reflect their intrinsic properties, i.e. line thickness and slant. 25

4.1 Two-dimensional linear embedding of face images by Laplacianfaces. As can be seen, the face images are divided into two parts, the faces with open mouth and the faces with closed mouth. Moreover, it can be clearly seen that the pose and expression of human faces change continuously and smoothly, from top to bottom, from left to right. The bottom images correspond to points along the right path (linked by solid line), illustrating one particular mode of variability in pose. . 42

4.2 Distribution of the 10 testing samples in the reduced representation subspace. As can be seen, these testing samples optimally find their coordinates which reflect their intrinsic properties, i.e. pose and expression. 43

4.3 The eigenvalues of LPP and Laplacian Eigenmap. 44
4.4 The first 10 Eigenfaces (first row), Fisherfaces (second row) and Laplacianfaces (third row) calculated from the face images in the YALE database. 45
4.5 The original face image and the cropped image. 46
4.6 Sample face images of an individual from Yale face database. 46
4.7 Recognition accuracy vs. dimensionality reduction on Yale database. 48
4.8 The sample cropped face images of one individual from PIE database. The original face images are taken under varying pose, illumination, and expression. 48
4.9 Recognition accuracy vs. dimensionality reduction on PIE database. . 49
4.10 The sample cropped face images of 8 individuals from MSRA database. The face images in the first row are taken in the first session, which are used for training. The face images in the second row are taken in the second session, which are used for testing. The two images in the same column are corresponding to the same individual. 50
4.11 Recognition accuracy vs. dimensionality reduction on MSRA database. 51
4.12 Performance comparison on the MSRA database with different number of traning samples. The training samples are from the set MSRA-S1. The images in MSRA-S2 are used for testing exclusively. As can be seen, our Laplacianfaces method takes advantage of more training samples. The error rate reduces from 26.04% with 10% training samples to 8.17% with 100% training samples. There is no evidence for Fisherfaces that it can take advantage of more training samples. 52

5.1 (Evaluation on similarity measure) The overall average precision using the Reuters-21578 document corpus. 67
5.2 (Evaluation on similarity measure) The optimal dimensions of the baseline, LSI and LPI algorithms using the Reuters-21578 document corpus. 67
5.3 (Evaluation on similarity measure) The overall average precision using the TDT2 document corpus. 70
5.4 (Evaluation on similarity measure) The optimal dimensions of the baseline, LSI and LPI algorithms using the TDT2 document corpus. . . . 70
5.5 (Evaluation on discriminating power) The overall average accuracies of the baseline, LSI and LPI algorithms using the Reuters-21578 document corpus. 73
5.6 (Evaluation on discriminating power) The optimal dimensions of the baseline, LSI and LPI algorithms using the Reuters-21578 document corpus. 73
5.7 Discriminating power of LPI VS. the number of nearest neighbors using the Reuters-21578 document corpus. 75

5.8 (Evaluation on discriminating power) The overall average accuracies of the baseline, LSI and LPI algorithms using the TDT2 document corpus. 76
5.9 (Evaluation on discriminating power) The optimal dimensions of the baseline, LSI and LPI algorithms using the TDT2 document corpus. . 77
5.10 Discriminating power of LPI VS. the number of nearest neighbors using the TDT2 document corpus. 77
5.11 2D Embedding results of LSI and LPI on the TDT2 corpus. Each color represents a topic. 80
5.12 2D Embedding results of LSI and LPI on the TDT2 corpus. Each color represents a topic. 81
5.13 Optimal dimension with different number of classes on TDT2 corpus. Each bar shows the average of 50 test runs, and the error bar indicates the standard deviation. 83
5.14 Optimal dimension with different number of classes on Reuters Corpus. Each bar shows the average of 50 test runs, and the error bar indicates the standard deviation. 84

6.1 2-D visualization of the Iris data. 97
6.2 Sample face images from the CMU PIE face database. For each subject, there are 24 face images under different lighting conditions with fixed pose and expression. 98
6.3 Clustering results on 5 randomly chosen classes. 101
6.4 Clustering results on 10 randomly chosen classes. 102
6.5 Clustering results on 30 randomly chosen classes. 103
6.6 Clustering results on 68 classes. 104

7.1 Error rate vs. dimensionality reduction on PIE database 118
7.2 Error rate vs. dimensionality reduction on ORL database 120
7.3 2D visualization of image set 123
7.4 Sample face images from CMU PIE database. For each subject, there are 22 face images under different lighting conditions and fixed pose (C27) and expression. 124
7.5 Clustering results on 5 randomly chosen classes 125
7.6 Clustering results on 10 randomly chosen classes 126
7.7 Clustering results on 30 randomly chosen classes 126
7.8 Clustering results on 68 randomly chosen classes 127

LIST OF TABLES

2.1 20 newsgroups classification accuracy with 6,000 training documents . 26
2.2 20 newsgroups classification accuracy with 10,000 training documents 26
2.3 20 newsgroups classification accuracy with 15,000 training documents 27

4.1 Performance comparison on the Yale database 47
4.2 Performance comparison on the PIE database 49
4.3 Performance comparison on the MSRA database 50
4.4 Recognition error rate with different number of training samples . . . 51
4.5 Performance comparisons on CMU PIE face database 53

5.1 30 semantic categories from Reuters-21578 used in our experiments . 64
5.2 30 semantic categories from TDT2 used in our experiments 64
5.3 Evaluation on similarity measure using the Reuters-21578 document corpus. The first column contains the keywords that are used to generate the document subsets. The LPI, LSI and baseline algorithms are compared. "AvP" stands for the average precision and "DR rate" stands for the dimensionality reduction rate. 68
5.4 Evaluation on similarity measure using the TDT2 document corpus. The first column contains the keywords that are used to generate the document subsets. The LPI, LSI and baseline algorithms are compared. "AvP" stands for the average precision and "DR rate" stands for the dimensionality reduction rate. 71
5.5 Evaluation on discriminating power using the Reuters-21578 document corpus. The first column contains the keywords that are used to generate the document subsets. The LPI, LSI and baseline algorithms are compared. "AvP" stands for the average precision and "DR rate" stands for the dimensionality reduction rate. 74
5.6 Evaluation on discriminating power using the TDT2 document corpus. The first column contains the keywords that are used to generate the document subsets. The LPI, LSI and baseline algorithms are compared. "AvP" stands for the average precision and "DR rate" stands for the dimensionality reduction rate. 78
5.7 Performance comparisons on TDT2 corpus 82
5.8 Performance comparisons on Reuters corpus 85

6.1 Accuracy comparisons (k is the number of clusters) 100
6.2 Mutual Information comparisons (k is the number of clusters) 105

7.1 Performance comparison on PIE database 119
7.2 Performance comparison on ORL database 121
7.3 Clustering performance comparisons on CMU PIE database 124

CHAPTER 1
INTRODUCTION

Suppose we have a collection of data points of n-dimensional real vectors drawn from an unknown probability distribution. In increasingly many cases of interest in machine learning and data mining, one is confronted with the situation where n is *very large*. However, there might be reason to suspect that the "intrinsic dimensionality" of the data is much lower. This leads one to consider methods of dimensionality reduction that allow one to represent the data in a lower dimensional space. Take the following typical cases: a face recognition system based on $m \times n$ greyscale images which, by row concatenation, can be transformed into $mn-$dimensional real vectors. In practice, one could have images of $m = n = 256$, or 65536-dimensional vectors; if, say, a multilayer perceptron was to be used as the classification system, the number of weights would be exceedingly large and would require an enormous training set to avoid overfitting. Therefore we need to reduce the dimension.

A great number of dimensionality reduction techniques exist in the literature. In practical situations, when n is prohibitively large, one is often forced to use linear techniques. Consequently, projective maps have been the subject of considerable investigation. Two classical yet popular forms of linear techniques are the methods of Principal Component Analysis (PCA) [40] and Linear Discriminant Analysis (LDA) [22]. PCA and LDA are eigenvector methods designed to model linear variabilities in high-dimensional data.

Recently, there has been renewed interest in the problem of developing low dimensional representations when data arises from sampling a probability distribution on a low dimensional manifold embedded in the high dimensional ambient space [10],[15],[61],[71],[86]. These methods yield impressive results on some benchmark artificial data sets, as well as on real world data sets. However, their nonlinear property makes them computationally expensive. Moreover, they often yield maps that are defined only on the *training* data points and how to evaluate the map on novel *test* points remains unclear.

Some other popular nonlinear techniques, like self-organizing maps and neural network based approaches [32], set up a nonlinear optimization problem whose solution is typically obtained by gradient descent that is only guaranteed to produce a local optimum. Kernel method have recently been used to generalize PCA and LDA to nonlinear maps [64],[8]. However, these algorithms do not explicitly consider the structure of the manifold on which the data may possibly reside.

This thesis starts with the assumption that the data lie on or around a low-dimensional manifold in a high-dimensional Euclidean space. We then introduce a set of data representation algorithms including Locality Preserving Projection, Laplacian Score and Tensor Subspace Analysis.

CHAPTER 2
LOCALITY PRESERVING PROJECTIONS

2.1 Linear Techniques for Dimensionality Reduction

One approach to coping with the problem of excessive dimensionality of data feature space is to reduce the dimensionality by combining features. Linear combinations are particular attractive because they are simple to compute and analytically tractable. In effect, linear methods project the high-dimensional data onto a lower dimensional subspace.

Consider the problem of representing all of the vectors in a set of m n-dimensional samples $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$, with zero mean, by a single vector $\mathbf{y} = \{y_1, y_2, \cdots, y_m\}$ such that y_i *represent* $\mathbf{x}_i$. Specifically, we find a linear mapping from the n-dimensional space to a line. Without loss of generality, we denote the transformation vector by $\mathbf{a}$. That is, $\mathbf{a}^T\mathbf{x}_i = y_i$. Actually, the magnitude of $\mathbf{a}$ is of no real significance, because it merely scales y_i. Different objective functions will yield different algorithms with different properties.

PCA seeks a projection that best represents the data in a least-squares sense. The matrix $\mathbf{a}\mathbf{a}^T$ is a projection onto the principal component space spanned by $\mathbf{a}$ which minimizes the following objective function,

$$\min_{\mathbf{a}} \sum_{i=1}^{m} |\mathbf{x}_i - \mathbf{a}\mathbf{a}^T\mathbf{x}_i|^2 \tag{2.1}$$

The output set of principal vectors $\mathbf{a}_1, \mathbf{a}_1, \cdots, \mathbf{a}_l$ are an orthonormal set of vectors representing the eigenvectors of the sample covariance matrix associated with the l largest eigenvalues.

While PCA seeks directions that are efficient for representation, Linear Discriminant Analysis seeks directions that are efficient for discrimination. Suppose we have a set of m n-dimensional samples $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$, belonging to l classes of faces. The objective function is as follows,

$$\max_{\mathbf{a}} \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T S_w \mathbf{a}} \tag{2.2}$$

$$S_b = \sum_{i=1}^{l} |C_i| (\mu^i - \mu)(\mu^i - \mu)^T \tag{2.3}$$

$$S_w = \sum_{i=1}^{l} |C_i| E\left[(\mathbf{x}^i - \mu^i)(\mathbf{x}^i - \mu^i)^T\right] \tag{2.4}$$

where μ is the total sample mean vector, $|C_i|$ is the number of samples in class C_i, μ^i are the average vectors of C_i, and $\mathbf{x}^i$ are the sample vectors associated to C_i. We call S_w the $within-class\ scatter\ matrix$ and S_b the $between-class\ scatter\ matrix$.

Some other linear techniques for dimensionality reduction are proposed recently, e.g. random projection [19], informed projection [18], region-based PCA [48], etc. However, none of them explicitly considers the manifold structure.

2.2 The Algorithm

In this section, we introduce a new linear dimensionality reduction algorithm — Locality Preserving Projections (LPP). The primary consideration of LPP is to preserve the neighborhood structure of the data set.

The generic problem of linear dimensionality reduction is the following. Given a set $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$ in $\mathbf{R}^n$, find a transformation matrix A that maps these m points to a set of points $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_m$ in $\mathbf{R}^l$ ($l \ll n$), such that $\mathbf{y}_i$ "represents" $\mathbf{x}_i$, where $\mathbf{y}_i = A^T\mathbf{x}_i$.

Our method is of particular applicability in the special case where $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m \in \mathcal{M}$ and $\mathcal{M}$ is a nonlinear manifold embedded in $\mathbf{R}^n$. The algorithmic procedure is formally stated below:

1. **Constructing the adjacency graph**: Let G denote a graph with m nodes.

We put an edge between nodes i and j if $\mathbf{x}_i$ and $\mathbf{x}_j$ are "close". There are two variations:

(a) ϵ-neighborhoods. [parameter $\epsilon \in \mathbf{R}$] Nodes i and j are connected by an edge if $\|\mathbf{x}_i - \mathbf{x}_j\|^2 < \epsilon$ where the norm is the usual Euclidean norm in $\mathbf{R}^n$.

(b) k nearest neighbors. [parameter $k \in \mathbf{N}$] Nodes i and j are connected by an edge if i is among k nearest neighbors of j or j is among k nearest neighbors of i.

Note: The method of constructing an adjacency graph outlined above is correct if the data actually lie on a low dimensional manifold. In general, however, one might take a more utilitarian perspective and construct an adjacency graph based on any principle (for example, perceptual similarity for natural signals, hyperlink structures for web documents, etc.). Once such an adjacency graph is obtained, LPP will try to optimally preserve it in choosing projections.

2. **Choosing the weights**: Here, as well, we have two variations for weighting the edges. W is a sparse symmetric $m \times m$ matrix with W_{ij} having the weight of the edge joining vertices i and j, and 0 if there is no such edge.

   (a) Heat kernel. [parameter $t \in \mathbf{R}$]. If nodes i and j are connected, put

$$W_{ij} = exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/t) \tag{2.5}$$

   The justification for this choice of weights can be traced back to [1].

   (b) Simple-minded. [No parameter]. $W_{ij} = 1$ if and only if vertices i and j are connected by an edge.

3. **Eigenmaps**: Compute the eigenvectors and eigenvalues for the generalized eigenvector problem:

$$XLX^T\mathbf{a} = \lambda XDX^T\mathbf{a} \tag{2.6}$$

where D is a diagonal matrix whose entries are column (or row, since W is symmetric) sums of W, $D_{ii} = \Sigma_j W_{ji}$. $L = D - W$ is the Laplacian matrix. The

i^{th} column of matrix X is $\mathbf{x}_i$.

Let the column vectors $\mathbf{a}_0, \mathbf{a}_1, \cdots, \mathbf{a}_{l-1}$ be the solutions of equation (1), ordered according to their eigenvalues, $\lambda_0 < \lambda_1 < \cdots < \lambda_{l-1}$. Thus, the embedding is as follows:

$$\mathbf{x}_i \rightarrow \mathbf{y}_i = A^T \mathbf{x}_i \tag{2.7}$$

$$A = \begin{pmatrix} \mathbf{a}_0, & \mathbf{a}_1, & \cdots, & \mathbf{a}_{l-1} \end{pmatrix} \tag{2.8}$$

where $\mathbf{y}_i$ is a l-dimensional vector, and A is a $n \times l$ matrix.

2.3 Theoretical Justifications

The Locality Preserving Projection algorithm is fundamentally based on Laplacian Eigenmaps [10]. Following the discussion in [10], we provide in this section a justification of the new linear dimensionality reduction algorithm.

2.3.1 Optimal Linear Embedding

The following section is based on standard spectral graph theory. See [17] for a comprehensive reference and [10] for applications to data representation.

Recall that given a data set we construct a weighted graph $G = (V, E)$ with edges connecting nearby points to each other. Consider the problem of mapping the weighted graph G to a line so that connected points stay as close together as possible. Let $\mathbf{y} = (y_1, y_2, \cdots, y_m)^T$ be such a map. A reasonable criterion for choosing a "good" map is to minimize the following objective function [10]

$$\sum_{ij} (y_i - y_j)^2 W_{ij} \tag{2.9}$$

under appropriate constraints. The objective function with our choice of weights W_{ij} incurs a heavy penalty if neighboring points $\mathbf{x}_i$ and $\mathbf{x}_j$ are mapped far apart.

Therefore, minimizing it is an attempt to ensure that if $\mathbf{x}_i$ and $\mathbf{x}_j$ are "close" then y_i and y_j are close as well.

If we restrict the map to be linear, i.e. $y_i = \mathbf{a}^T\mathbf{x}_i$, then we can reduce the above objective function as follows:

$$
\begin{aligned}
& \frac{1}{2}\sum_{ij}(y_i - y_j)^2 W_{ij} && (2.10)\\
= & \frac{1}{2}\sum_{ij}(\mathbf{a}^T\mathbf{x}_i - \mathbf{a}^T\mathbf{x}_j)^2 W_{ij} && (2.11)\\
= & \sum_{i}\mathbf{a}^T\mathbf{x}_i D_{ii}\mathbf{x}_i^T\mathbf{a} - \sum_{ij}\mathbf{a}^T\mathbf{x}_i W_{ij}\mathbf{x}_i^T\mathbf{a} && (2.12)\\
= & \mathbf{a}^T X(D-W)X^T\mathbf{a} && (2.13)\\
= & \mathbf{a}^T XLX^T\mathbf{a} && (2.14)
\end{aligned}
$$

where $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m]$, and D is a diagonal matrix; its entries are column (or row, since W is symmetric) sum of W, $D_{ii} = \Sigma_j W_{ij}$. $L = D - W$ is the Laplacian matrix [17]. Matrix D provides a natural measure on the data points. The bigger the value D_{ii} (corresponding to y_i) is, the more "important" is y_i. Therefore, we impose a constraint as follows:

$$\mathbf{y}^T D\mathbf{y} = 1 \Rightarrow \mathbf{a}^T XDX^T\mathbf{a} = 1 \tag{2.15}$$

Thus, the minimization problem reduces to finding

$$\underset{\substack{\mathbf{a} \\ \mathbf{a}^T XDX^T\mathbf{a}=1}}{\arg\min} \quad \mathbf{a}^T XLX^T\mathbf{a} \tag{2.16}$$

The constraint $\mathbf{y}^T D\mathbf{y} = 1$ removes an arbitrary scaling factor in the embedding. The Laplacian matrix is a symmetric, positive semi-definite matrix which can be thought of as an operator on functions defined on vertices of G. The close relationship between the graph Laplacian and the Laplace Beltrami operator on the manifold is discussed in [17][10].

We will now switch to a Lagrangian formulation of the problem. The Lagrangian is as follows

$$\mathcal{L} = \mathbf{a}^T XLX^T\mathbf{a} - \lambda\mathbf{a}^T XDX^T\mathbf{a} \tag{2.17}$$

Requiring that the gradient of $\mathcal{L}$ vanish gives the following eigenvector problem:

$$XLX^T\mathbf{a} = \lambda XDX^T\mathbf{a} \tag{2.18}$$

It is easy to show that the matrices XLX^T and XDX^T are both symmetric and positive semi-definite. The vectors $\mathbf{a}_i(i = 0, 2, \cdots, l-1)$ that minimize the objective function are given by the minimum eigenvalue solutions to the generalized eigenvalue problem.

Note that XLX^T and XDX^T are two $n\times n$ matrices, where n is the dimensionality of the input space. While in the nonlinear Laplacian Eigenmaps, L and D are two $m \times m$ matrices, where m is the number of data points in the data set. In most real world applications, $m \gg n$. This property makes LPP much more computationally tractable than the nonlinear Laplacian Eigenmaps.

2.3.2 Geometrical Justification

The Laplacian matrix L ($=D - W$) for finite graph, or *graph Laplacian* [17] [30] [54], is analogous to the Laplace Beltrami operator $\mathcal{L}$ on compact Riemannian manifolds. While the Laplace Beltrami operator for a manifold is generated by the Riemannian metric, for a graph it comes from the adjacency relation. It is important to note that the graph Laplacian is also widely used in the spectral clustering techniques [56] [68] [77].

Let $\mathcal{M}$ be a smooth, compact, d-dimensional Riemannian manifold. If the manifold is embedded in $\mathbf{R}^n$ the Riemannian structure on the manifold is induced by the standard Riemannian structure on $\mathbf{R}^n$. We are looking here for a map from the manifold to the real line such that points close together on the manifold get mapped close together on the line. Let f be such a map. Assume that $f : \mathcal{M} \to \mathbf{R}$ is twice differentiable.

Belkin and Niyogi [10] showed that the optimal map preserving locality can be found by solving the following optimization problem on the manifold:

$$\arg \min_{\|f\|_{L^2(\mathcal{M})}=1} \int_{\mathcal{M}} \|\nabla f\|^2 \tag{2.19}$$

which is equivalent to [1]

$$\arg \min_{\|f\|_{L^2(\mathcal{M})}=1} \int_{\mathcal{M}} \mathcal{L}(f) f \tag{2.20}$$

where the integral is taken with respect to the standard measure on a Riemannian manifold. $\mathcal{L}$ is the Laplace Beltrami operator on the manifold, i.e. $\mathcal{L}f = -\operatorname{div} \nabla(f)$. Thus, the optimal f has to be an eigenfunction of $\mathcal{L}$.

By the standard spectral graph theory [17], minimizing the integral $\int_{\mathcal{M}} \mathcal{L}(f)f$ corresponds analogously to minimizing $\langle f(X), Lf(X)\rangle = f^T(X)Lf(X)$ on a graph, where

$$X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m], \mathbf{x}_i \in \mathcal{M} \tag{2.21}$$

$$f(X) = [f(\mathbf{x}_1), f(\mathbf{x}_2, \cdots, f(\mathbf{x}_m))]^T \tag{2.22}$$

$$f^T(X) = [f(\mathbf{x}_1), f(\mathbf{x}_2, \cdots, f(\mathbf{x}_m))] \tag{2.23}$$

If we restrict the map to be linear, i.e. $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x}$, then we have

$$f(X) = X^T\mathbf{a} \tag{2.24}$$

$$\langle f(X), Lf(X)\rangle = f^T(X)Lf(X) = \mathbf{a}^T XLX^T\mathbf{a} \tag{2.25}$$

[1]If $\mathcal{M}$ has a boundary, appropriate boundary conditions for f need to be assumed.

The constraint can be computed as follows,

$$\begin{aligned}
& \|f\|^2_{L^2(\mathcal{M})} && (2.26)\\
= & \int_{\mathcal{M}} |f(\mathbf{x})|^2 d\mathbf{x} && (2.27)\\
= & \int_{\mathcal{M}} (\mathbf{a}^T\mathbf{x})^2 d\mathbf{x} && (2.28)\\
= & \int_{\mathcal{M}} (\mathbf{a}^T\mathbf{x}\mathbf{x}^T\mathbf{a}) d\mathbf{x} && (2.29)\\
= & \mathbf{a}^T(\int_{\mathcal{M}} \mathbf{x}\mathbf{x}^T d\mathbf{x})\mathbf{a} && (2.30)
\end{aligned}$$

where $d\mathbf{x}$ is the standard measure on a Riemannian manifold. Again, by spectral graph theory [17], the measure $d\mathbf{x}$ directly corresponds to the measure for the graph which is the degree of the vertex, i.e. D_{ii}. Thus, the $\mathcal{L}^2$ norm of f can be discretely approximated as follows,

$$\begin{aligned}
& \|f\|^2_{L^2(\mathcal{M})} && (2.31)\\
= & \mathbf{a}^T(\int_{\mathcal{M}} \mathbf{x}\mathbf{x}^T d\mathbf{x})\mathbf{a} && (2.32)\\
\approx & \mathbf{a}^T(\sum_i \mathbf{x}_i\mathbf{x}_i^T D_{ii})\mathbf{a} && (2.33)\\
= & \mathbf{a}^T X D X^T \mathbf{a} && (2.34)
\end{aligned}$$

Finally, we conclude that the optimal linear projective map, i.e. $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x}$, can be obtained by solving the following objective function,

$$\begin{aligned}
\arg \min_{\mathbf{a}} \quad & \mathbf{a}^T X L X^T \mathbf{a} && (2.35)\\
\mathbf{a}^T X D X^T \mathbf{a} = 1 &
\end{aligned}$$

These projective maps are the optimal linear approximations to the eigenfunctions of the Laplace Beltrami operator on the manifold. Therefore, they are capable of discovering the nonlinear manifold structure.

2.4 Connections to Principal Component Analysis

In this section, we show that PCA can be obtained from LPP with different choice of graph structure.

First, it would be important to note that XLX^T is the data covariance matrix, if the Laplacian matrix L is $\frac{1}{m}I - \frac{1}{m^2}\mathbf{e}\mathbf{e}^T$, where m is the number of data points, I is the identity matrix and $\mathbf{e}$ is a column vector taking 1 at each entry. In fact, the Laplacian matrix here has the effect of removing the sample mean from the sample vectors. In this case, the weight matrix W takes $\frac{1}{m^2}$ at each entry, i.e. $W_{ij} = \frac{1}{m^2}$, $\forall i, j$. $D_{ii} = \sum_j W_{ij} = \frac{1}{m}$. Hence the Laplacian matrix is $L = D - W = \frac{1}{m}I - \frac{1}{m^2}\mathbf{e}\mathbf{e}^T$. Let μ denote the sample mean, i.e. $\mu = \frac{1}{m}\sum_i \mathbf{x}_i$. Thus, it is easy to check that $XLX^T = \frac{1}{m}\sum_i(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T$, which is the data covariance matrix.

The above analysis shows that the weight matrix W plays a key role in the LPP algorithm. When we aim at preserving the global structure, we take ϵ (or k) to be infinity and choose the eigenvectors (of the matrix XLX^T) associated with the largest eigenvalues. Hence the data points are projected along the direction of maximal variance.

2.5 Connections to Linear Discriminant Analysis

Linear Discriminant Analysis (LPP) can be also obtained in our LPP framework with different choice of graph structure. Recall that LDA seeks directions that are efficient for discrimination. The projection is found by solving the generalized eigenvalue problem:

$$S_b\mathbf{a} = \lambda S_w \mathbf{a} \tag{2.36}$$

where S_b and S_w are defined in Section 2. Suppose there are l classes. The i^{th} class contains m_i sample points. Let μ denote the average vector of the whole data set and μ^i denote the average vector of the i^{th} class. Let $\mathbf{x}_j^i$ denote the j^{th} sample point in the i^{th} class. We can rewrite the matrix S_w as follows:

$$\begin{aligned}
S_w &= \sum_{i=1}^{l}\left(\sum_{j=1}^{m_i}\left(\mathbf{x}_j^i - \mu^i\right)\left(\mathbf{x}_j^i - \mu^i\right)^T\right)\\
&= \sum_{i=1}^{l}\left(\sum_{j=1}^{m_i}\left(\mathbf{x}_j^i(\mathbf{x}_j^i)^T - \mu^i(\mathbf{x}_j^i)^T - \mathbf{x}_j^i(\mu^i)^T + \mu^i(\mu^i)^T\right)\right)\\
&= \sum_{i=1}^{l}\left(\sum_{j=1}^{m_i}\mathbf{x}_j^i(\mathbf{x}_j^i)^T - m_i\mu^i(\mu^i)^T\right)\\
&= \sum_{i=1}^{l}\left(X_iX_i^T - \frac{1}{m_i}\left(\mathbf{x}_1^i + \cdots + \mathbf{x}_{m_i}^i\right)\left(\mathbf{x}_1^i + \cdots + \mathbf{x}_{m_i}^i\right)^T\right)\\
&= \sum_{i=1}^{l}\left(X_iX_i^T - \frac{1}{m_i}X_i\left(\mathbf{e}_i\mathbf{e}_i^T\right)X_i^T\right)\\
&= \sum_{i=1}^{l}X_i\left(I_i - \frac{1}{m_i}\mathbf{e}_i\mathbf{e}_i^T\right)X_i^T\\
&= \sum_{i=1}^{l}X_iL_iX_i^T
\end{aligned}$$

where $X_iL_iX_i^T$ is the data covariance matrix of the i^{th} class. To further simplify the above equation, we define:

$$W_{ij} = \begin{cases} \frac{1}{m_t}, & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ both belong to the } t^{th} \text{ class;} \\ 0, & \text{otherwise.} \end{cases} \tag{2.37}$$

$$L = I - W \tag{2.38}$$

where I is a $m \times m$ identity matrix. Thus, we get:

$$S_w = XLX^T \tag{2.39}$$

Similarly, we can compute S_b as follows:

$$\begin{aligned}
S_b &= \sum_{i=1}^{l} m_i \left(\mu^i - \mu\right)\left(\mu^i - \mu\right)^T \\
&= \left(\sum_{i=1}^{l} m_i \mu^i \left(\mu^i\right)^T\right) - \mu \left(\sum_{i=1}^{l} m_i \left(\mu^i\right)^T\right) - \left(\sum_{i=1}^{l} m_i \left(\mu^i\right)\right)\mu^T + m\mu\mu^T \\
&= \left(\sum_{i=1}^{l} \frac{1}{m_i}\left(\mathbf{x}_1^i + \cdots + \mathbf{x}_{m_i}^i\right)\left(\mathbf{x}_1^i + \cdots + \mathbf{x}_{m_i}^i\right)^T\right) - m\mu\mu^T - m\mu\mu^T + m\mu\mu^T \\
&= \left(\sum_{i=1}^{l} \sum_{j,k=1}^{m_i} \frac{1}{m_i}\mathbf{x}_j^i(\mathbf{x}_k^i)^T\right) - m\mu\mu^T \\
&= XWX^T - m\mu\mu^T \\
&= XWX^T - X(\frac{1}{m}\mathbf{e}\mathbf{e}^T)X^T \\
&= X(W - I + I - \frac{1}{m}\mathbf{e}\mathbf{e}^T)X^T \\
&= -XLX^T + X(I - \frac{1}{m}\mathbf{e}\mathbf{e}^T)X^T \\
&= -XLX^T + C
\end{aligned}$$

where $C = X(I - \frac{1}{n}\mathbf{e}\mathbf{e}^T)X^T$ is the data covariance matrix. Thus, the generalized eigenvector problem of LDA can be written as follows:

$$\begin{aligned}
& S_b\mathbf{a} = \lambda S_w \mathbf{a} \\
\Rightarrow \quad & (C - XLX^T)\mathbf{a} = \lambda XLX^T\mathbf{a} \\
\Rightarrow \quad & C\mathbf{a} = (1+\lambda)XLX^T\mathbf{a} \\
\Rightarrow \quad & XLX^T\mathbf{a} = \frac{1}{1+\lambda}C\mathbf{a}
\end{aligned}$$

which is equivalent to the minimum eigenvalue solutions to the following eigenproblem:

$$XLX^T\mathbf{a} = \lambda C\mathbf{a} \tag{2.40}$$

With the weight matrix defined in 2.37, we have $D = I$ and $XDX^T = XX^T$. If the sample mean μ is a zero vector, XX^T is right the covariance matrix, $XX^T = C$.

This shows that, LDA can be induced from the LPP framework with a certain weight matrix.

2.6 Computational Issues

As described in Section 3, we need to solve a generalized eigenvalue problem, $L'\mathbf{a} = \lambda D'\mathbf{a}$, where $L' = XLX^T$ and $D' = XDX^T$. If the matrix D' is invertible, then the generalized eigenvalue problem can be transformed to a ordinary eigenvalue problem, i.e. $D^{'-1}L'\mathbf{a} = \lambda\mathbf{a}$. However, the eigensystem computation of the ordinary eigenvalue problem could be unstable since the matrix $D^{'-1}L'$ may not to be symmetric. To avoid this problem, the following method diagonalizes the symmetric matrix, and produces a stable eigensystem computation procedure.

By applying eigenvector decomposition, D' can be decomposed into the product of three matrices as follows:

$$D' = USU^T \tag{2.41}$$

where $S = diag\{\lambda_1, \lambda_2, \cdots, \lambda_l\}$ are the eigenvalues and $U = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_l]$ are the corresponding eigenvectors of D'. U is othonormal in that $U^TU = I$ and $UU^T = I$. Since D' is positive semi-definite, $\lambda_i \geq 0, i = 1, 2, \cdots, l$. Now, we can rewrite equation (2.18) as follows:

$$\begin{aligned} & L'\mathbf{a} = \lambda D'\mathbf{a} && (2.42)\\ \Longrightarrow\quad & L'\mathbf{a} = \lambda USU^T\mathbf{a} && (2.43)\\ \Longrightarrow\quad & U^TL'U(U^T\mathbf{a}) = \lambda S(U^T\mathbf{a}) && (2.44)\end{aligned}$$

If D' is non-singular, then each diagonal entry of S is greater than zero (D' is positive semi-definite, hence its eigenvalues are real numbers, and no less than zero). In this case, $S^{-1/2}$ exists and can be easily obtained from S. If D' is singular, we define $S^{-1/2}$

as follows:

$$(S^{-1/2})_{ii} = \begin{cases} S_{ii}^{-1/2}, & \text{if } S_{ii} > 0; \\ 0, & \text{otherwise.} \end{cases} \tag{2.45}$$

In fact, $S^{-1/2}$ is a generalized inverse of $S^{1/2}$. Now, we have the following equation:

$$S^{-1/2}U^T L' U S^{-1/2}(S^{1/2}U^T\mathbf{a}) = \lambda S^{1/2}U^T\mathbf{a} \tag{2.46}$$

Let $W = S^{-1/2}U^T L' U S^{-1/2}$ and $\mathbf{b} = S^{1/2}U^T\mathbf{a}$. Finally, equation (70) can be simplified to solve the following eigenvector problem:

$$W\mathbf{b} = \lambda\mathbf{b} \tag{2.47}$$

where W is a symmetric matrix. Once we solve the symmetric eigenvalue problem (75) to obtain λ and $\mathbf{b}$, $\mathbf{a} = US^{-1/2}\mathbf{b}$ is the solution to the generalized eigenvalue problem (70). This is more stable and efficient than finding directly.

2.7 Model Selection for LPP

Typically the LPP algorithm have two tuning parameters, the number of nearest neighbors (k) and the dimensionality of the reduced space (d). The tuning parameter varies the complexity of our model, and we wish to find the values of k and d that minimize error.

The model selection methods can be either analytically based or resampling based. The typical analytically based methods include Akaike Information Criterion, Bayesian Information Criterion, and Minimum Description Length. The typical resampling based methods include Cross Validation and Bootstrap. We are particularly interested in resampling based methods since there is no explicit probabilistic formulation of the LPP algorithm and it is hard to apply the analytically based methods. Cross-Validation and Bootstrapping are both methods for estimating generalization error based on "resampling" [25], [35], [59], [67]. The resulting estimates of generalization error can be used for choosing among various models.

Ideally if we had enough data, we would set aside a validation set and use it to assess the performance of our model. Since data are often scarce, this is usually not possible. To solve this problem, M-fold cross-validation uses part of the available data to learn the model, and a different part to test it. We split the data into M roughly equal-sized parts. For the mth part, we learn the model to the other $M-1$ parts of the data, and calculate the prediction error of the learnt model when testing the mth part of the data. We do this for $m = 1, 2, \cdots, M$ and combine the M estimates of prediction error. Typical choices of M are 5 or 10 [31]. The case $M = N$ is known as *leave-one-out* cross-validation, where N is the number of data points.

The basic idea of bootstrap is to randomly draw datasets with replacement from the training data, each sample the same size as the original training set. This is done B times, producing B bootstrap datasets. Then we learn the model to each of the bootstrap datasets, and examine the behavior of the model over the B replications.

Sometimes applying model selection methods is time consuming. In the following we provide some empirical finding to help the choices of the parameters. In general, the best performance of LPP is obtained with small values of k (e.g. see figures 5.7 and 5.10). The optimal k is typically less than 10 in most situations. Also, from our experimental results on document clustering, it is clear that the performance of LPP is not very sensitive to the value of k. When LPP is applied to classification, k needs to be smaller than the number of training samples in a class. Otherwise, two samples sharing different labels might be linked together. For the optimal value of d, it is found that the optimal d should be close to the number of classes (c). This observation is supported by our experimental results on documents. Therefore, when the number of classes (c) is given, one can simply set the value of d to $c - 1$. This observation coincides with that in [56].

2.8 Complexity Analysis

In this Section, we provide a complexity analysis of the LPP algorithm. Basically, the complexity of LPP is dominated by two parts: k nearest neighbor search and solving a generalized eigenvector problem.

Consider m data points in n-dimensional space. For k nearest neighbor search, the complexity is $O((n+k)m^2)$. nm^2 stands for the complexity of computing the distances between any two data points. km^2 stands for the complexity of finding the k nearest neighbors for all the data points. The second part is solving the generalized eigenvector problem $A\mathbf{w} = \lambda B\mathbf{w}$, where A and B are $n \times n$ symmetric and positive semi-definite matrices. To solve this generalized eigenvector problem, we need first to computer the Singular Value Decomposition (SVD) of the matrix B. The complexity of SVD is $O(n^3)$. Then, to project the data points into a d-dimensional subspace, we need to compute the first d smallest eigenvectors of an $n \times n$ matrix, whose complexity is $O(dn^2)$. Thus, the total complexity of the generalized eigenvector problem is $O((n+d)n^2)$. Finally, we conclude that the time complexity of the LPP algorithm is $O((n+k)m^2 + (n+d)n^2)$. In general, k and d are much smaller than m and n. Therefore, the complexity of LPP is determined by the number of data points and the number of features.

For some problems of computer vision, such as face recognition, only a small number of data points are available. In such case, n is much smaller than m, and hence the complexity of the k nearest neighbor search is negligible comparing the generalized eigenvector problem. However, in information retrieval, m can be extremely large, ranging from thousands to millions. Thus, k nearest neighbor search becomes the bottleneck of the LPP algorithm. The canonical *nearest neighbor search (NNS)* problem can be formally stated below:

> Given a set of data points in n-dimensional space, we wish to preprocess these points into a data structure, so that given a query point, one can efficiently determine its nearest or generally k neighbor neighbors in the database.

Clearly, the algorithms for the NNS problem can be directly applied to our problem. We can simply take the query point as every data point in the database. There is a large literature on algorithms for the NNS problem, in both the exact and approximate cases [41]. Kleinberg proposed an algorithm which has query time $O(k+(n\log^2 n)(n+$

$\log m$)) and preprocessing $O((m \log n)^{2d})$ [41].

In our problem, the query points are restricted to be in the database. It turns out that this can be done in less time than the NNS problem. In this case, it is often referred to *all-nearest-neighbors (ANN) problem*. ANN can be formally stated below:

> Given a set $\mathcal{S}$ of m data points in n-dimensional space, find the nearest or generally k nearest neighbors in set $\mathcal{S}$ of each data point in $\mathcal{S}$.

Bentley has found an $O(m \log^{n-1} m)$ time algorithm for solving the ANN problem [12]. Vaidya has proposed an $O((cn)^n m \log m)$ algorithm [74], where c is constant independent to n and m. In the conventional nearest neighbor search problem, n is always assumed to be very small (< 10). Unfortunately, in many areas of pattern classification and information retrieval, n can be very large. For example, a typical face image is of size 32×32, which is in a 1024-dimensional space. It seems that the conventional algorithms for the NNS and ANN problems can not be applied.

2.9 Examples

In this section, we will present several examples to demonstrate how LPP works. We begin with three simple synthetic examples.

2.9.1 Simple synthetic Examples

Three toy examples were presented to show that LPP is less sensitive to outlier than PCA and can have more discriminating power than PCA in some cases. Moreover, LPP is capable of discovering the intrinsic topological structure of the data set. All of the three data sets correspond essentially to a one-dimensional manifold. Projection of the data points onto the first basis would then correspond to a one-dimensional linear manifold representation. The second basis, shown as a short line segment in the figure, would be discarded in this low-dimensional example.

Robustness

LPP is derived by preserving local information, hence it is less sensitive to outliers than PCA. This can be clearly seen from Figure 2.1. LPP finds the principal direc-

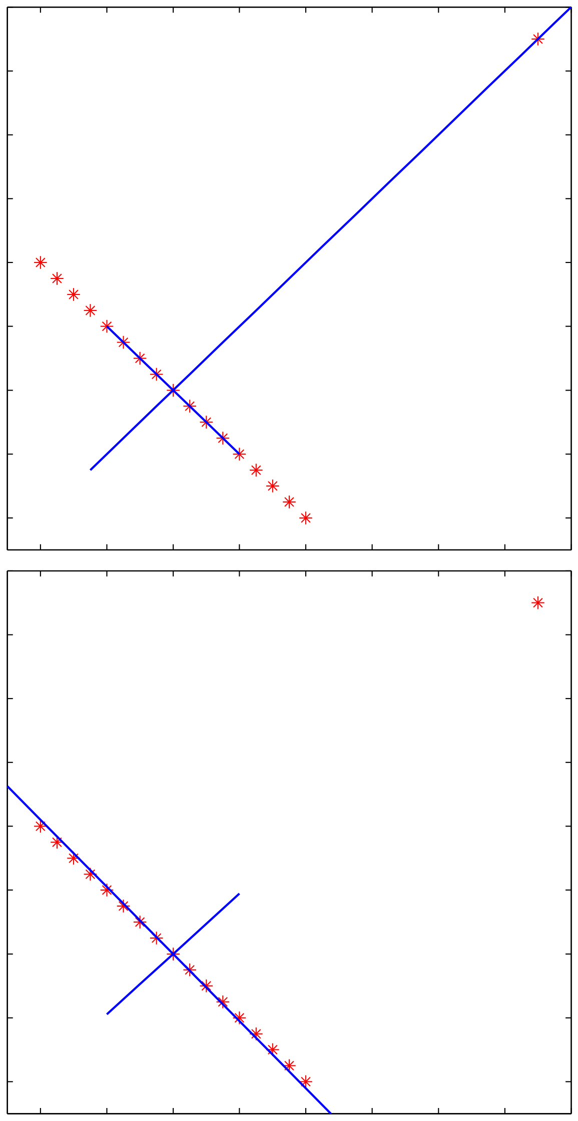


Figure 2.1: The upper plot shows the results of PCA. The lower plot shows the results of LPP. The straight line segments describe the two bases. The first basis is shown as a longer line segment, and the second basis is shown as a shorter line segment. Clearly, LPP is insensitive to the outlier at the upper right corner.

tion along the data points at the left bottom corner, while PCA finds the principal direction on which the data points at the left bottom corner collapse into a single point.

Discriminating Power

Moreover, LPP can have more discriminating power than PCA in some cases. As can be seen from Figure 2.2, the two circles are totally overlapped with each other in the principal direction obtained by PCA, while they are well separated in the principal direction obtained by LPP. Actually, in this example, LPP obtains the same result as LDA, if the two circles are viewed as two different classes of data points.

Topological Structure

The third data set is composed of three circles, as shown in Figure 2.3. By maximizing variance, PCA projects the three circles to three line segments. LPP projects the three circles to three circles in a plane while the local structure is preserved. As can seen from this example, LPP is capable of discovering the intrinsic topological structure of the data set while PCA fails.

2.9.2 2-D Data Visualization

An experiment was conducted with the *Multiple Features Database* [14]. This dataset consists of features of handwritten numbers (“0”-“9”) extracted from a collection of Dutch utility maps. 200 patterns per class (for a total of 2,000 patterns) have been digitized in binary images. Digits are represented in terms of Fourier coefficients, profile correlations, Karhunen-Love coefficients, pixel average, Zernike moments and morphological features. Each image is represented by a 649-dimensional vector. These data points are mapped to a 2-dimensional space using different dimensionality reduction algorithms, PCA, LPP, and Laplacian Eigenmaps. Note that, Kernel LPP yields the same results as Laplacian Eigenmaps since all the data points are used as training data. The experimental results are shown in Figure 2.4. As can be seen, LPP performs much better than PCA. It has much more discriminating power than PCA. As we described before, LPPs are obtained by finding the optimal linear approximations to the eigenfunctions of the Laplace Beltrami operator on the manifold. As a result,

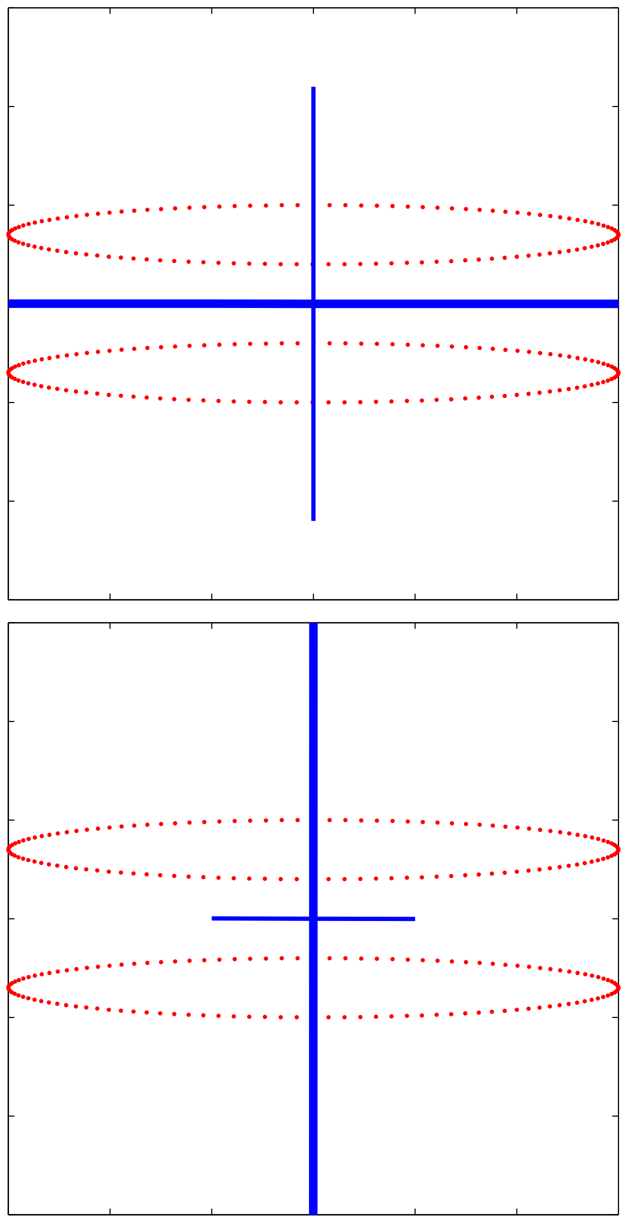


Figure 2.2: The upper plot shows the results of PCA. The lower plot shows the results of LPP. The straight line segments describe the two bases. The first basis is shown as a longer line segment, and the second basis is shown as a shorter line segment. In this example, LPP has more discriminating power than PCA.

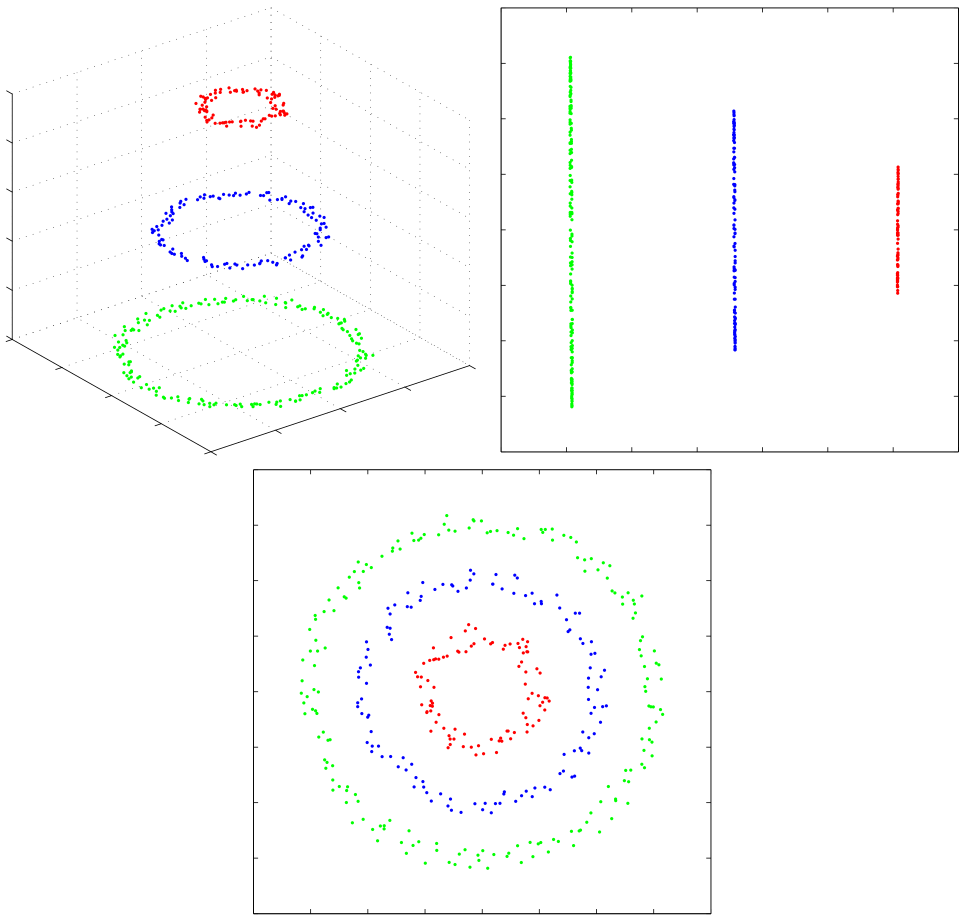


Figure 2.3: The first plot shows the data set which is composed of three circles. The second plot shows the 2-D projection by PCA. The third plot shows the 2-D projection by LPP. In this example, LPP is capable of discovering the intrinsic topological structure of the data set while PCA fails.

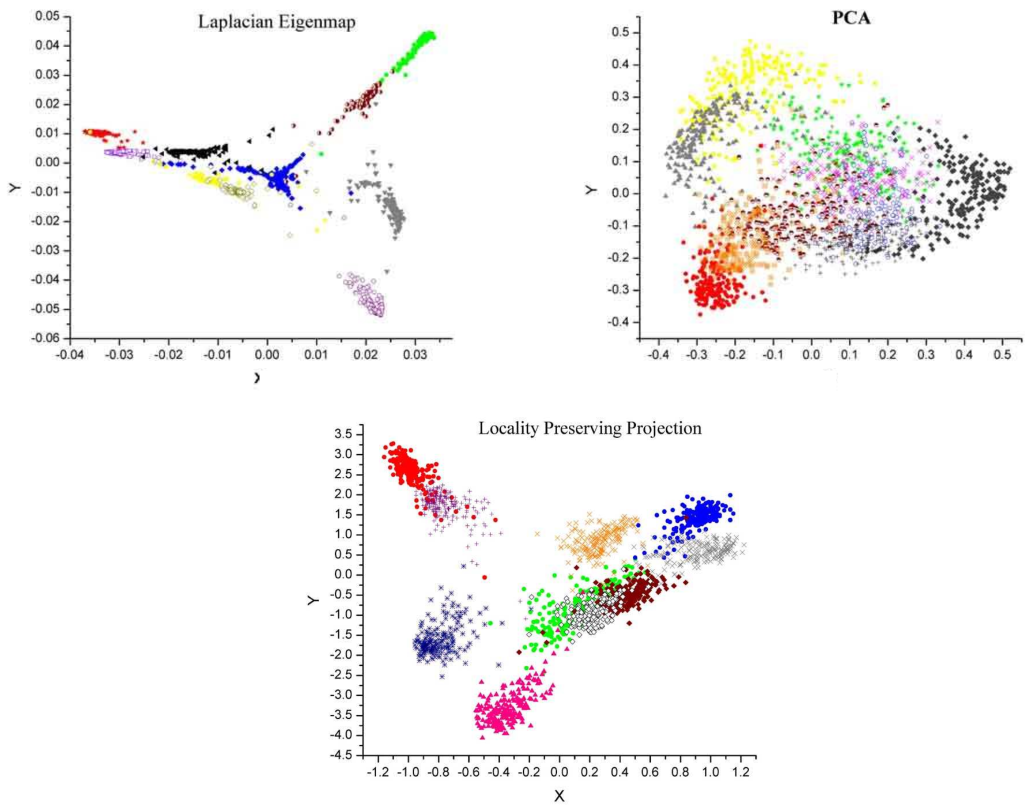


Figure 2.4: The handwritten digits (“0”-“9”) are mapped into a 2-dimensional space. The left panel in the first row is a representation of the set of all images of digits using the Laplacian eigenmap. The right panel in the first row shows the results of LPP using the first two directions to represent the data. The panel in the second row shows the results of principal component analysis. Each color corresponds to a digit.

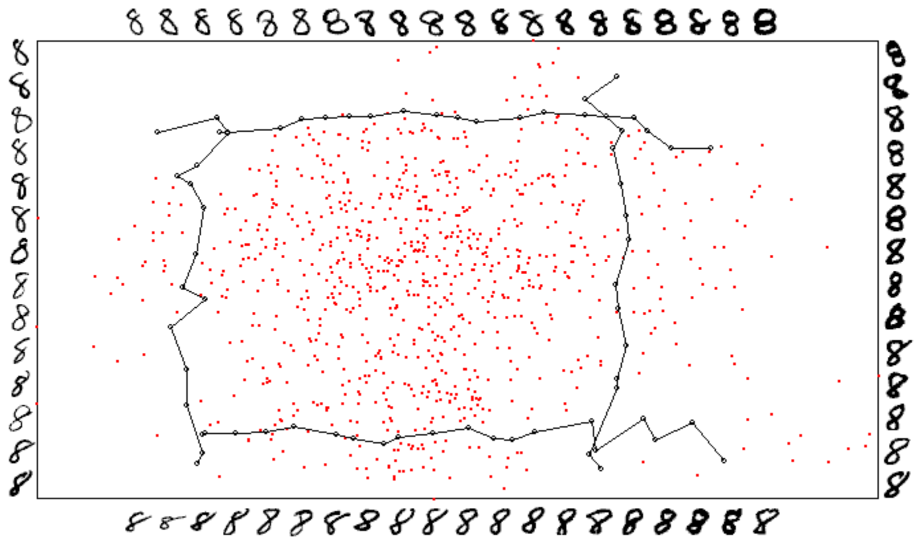


Figure 2.5: A two-dimensional representation of the set of images of handwritten digits using the Locality Preserving Projections. As can be seen, the first dimension (horizontal axis) appears to describe the line thickness of each digit, and the second dimension (vertical axis) appears to describe the slant of each digit.

LPP shares many of the data representation properties of non linear techniques such as Laplacian Eigenmap. However, LPP is computationally much more tractable.

2.9.3 Manifold of Handwritten Digits

We applied the LPP to sets of handwritten digit images which are believed to come from a manifold with few degrees of freedom. The data set is from the MNIST database. It consists of 974 examples of the digits ”8”. The size of each image is 28×28 pixels, with 256 gray levels per pixel. Thus, each image is represented by a 784-dimensional vector. We leave out 10 samples for testing, and the rest 964 samples are used to learn the projections by LPP. As can be seen in Figure 2.5, the digits are mapped into a two-dimensional space with continuous change in line thickness and slant. The first dimension (horizontal axis) appears to describe the line thickness of each digit, and the second dimension (vertical axis) appears to describe the slant of each digit. Some representative images are shown, which are sampled from left to right, and from top to bottom in the reduced representation space.

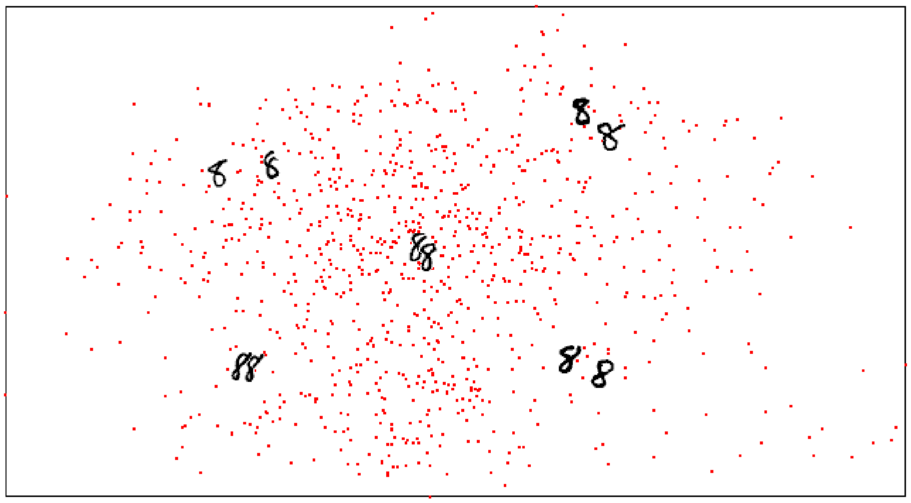


Figure 2.6: The distributions of 10 testing images in the 2-D plane. As can be seen, these testing samples optimally find their coordinates which reflect their intrinsic properties, i.e. line thickness and slant.

The ten testing samples can be simply located in the reduced space by the LPP transformation matrix. Figure 2.6 shows the result. As can be seen, these testing samples optimally find their coordinates which reflect their intrinsic properties, i.e. thickness and slant.

2.9.4 20 Newsgroups Classification

In this subsection, we consider the application of LPP on text classification. The text data set we used is the popular 20 Newsgroups data set. 20 Newsgroups is a data set collected and originally used for text classification by Lang [44]. It contains 19,997 documents evenly distributed across 20 classes. Each document is designated to one or more classes. Most of the documents have only one label, however some of them have two and more ones. In this paper, we ignored the problem of duplicated documents. The task is to identify to which newsgroup an document belongs to.

Using Rainbow, written by Andrew McCallum, we tokenized these 19,997 documents. We use the SMART system's stoplist to remove the general words, like "the" and "of", etc. The headers are also excluded, since they include the name of the

Table 2.1: 20 newsgroups classification accuracy with 6,000 training documents

dims	LPP (%) ($W_{ij} = 0/1$)	LPP (%) (heat kernel)	PCA+LPP (%) (heat kernel)	LSI (%)
100	44.2	46.2	51.9	45.0
200	52.3	52.5	55.2	50.0
300	54.4	54.7	55.5	51.5
400	54.7	55.3	55.8	52.8
500	54.9	55.2	55.6	53.7
600	55.0	55.1	55.2	54.2
700	54.4	54.5	54.7	54.7
800	54.0	54.2	54.6	55.1

Table 2.2: 20 newsgroups classification accuracy with 10,000 training documents

dims	LPP (%) ($W_{ij} = 0/1$)	LPP (%) (heat kernel)	PCA+LPP (%) (heat kernel)	LSI (%)
100	45.7	49.3	53.4	47.4
200	55.2	56.6	57.3	50.5
300	57.5	58.1	58.1	52.5
400	58.5	58.6	58.5	53.9
500	58.3	59.1	58.3	54.7
600	58.6	59.3	58.1	54.9
700	58.2	59.1	57.7	55.4
800	57.5	58.6	57.2	56.2
900	57.0	57.8	56.8	56.9
1000	56.6	57.2	56.6	57.3

Table 2.3: 20 newsgroups classification accuracy with 15,000 training documents

dims	LPP (%) ($W_{ij} = 0/1$)	LPP (%) (heat kernel)	PCA+LPP (%) (heat kernel)	LSI (%)
100	52.0	54.6	55.9	52.2
200	58.8	59.8	59.2	54.2
300	61.2	61.6	61.3	55.9
400	61.5	61.9	61.5	57.3
500	61.9	62.2	61.8	58.3
600	61.5	62.4	62.0	58.5
700	61.1	62.0	61.6	59.0
800	61.3	61.5	61.9	59.4
900	60.7	60.9	61.0	59.7
1000	60.1	60.5	60.5	60.1
1100	59.9	60.2	60.3	60.4

correct newsgroup. No further preprocessing is done. We kept the 2,000 words with highest class-dependent information gain. Each document is then represented by the counts of the most frequent 2000 words. Documents with 0 total count are removed, thus leaving us with 19,901 vectors in a 2000-dimensional space.

From the available 19,901 documents, we have created two different sets for our experiments: labelled documents for training, and unlabelled documents for testing. We average the results over 10 random splits. Three tests are performed. In the first test, the training set contains a total of 6000 documents, with 300 documents per class, and the remaining 13,901 documents form the testing set. In the second test, the training set contains a total of 10,000 documents, with 500 documents per class, and the remaining 9,901 documents form the testing set. In the third test, the training set contains a total of 15,000 documents, with 750 documents per class, and the remaining 4,901 documents form the testing set. In each test, the training set is used to learn a d-dimensional ($d < 2000$) linear subspace which preserves local information. We take the number of nearest neighbors for the algorithm to be 4, 5 and 6 for the three test, respectively. Both the training samples and the testing samples are mapped into the subspace. The nearest neighbor classifier is used for classification. The distance is taken to be the angle between the representation vectors. More sophisticated schemes, such as TF-IDF representations [5], increasing the weights of dimensions corresponding to more relevant words and treating cross-posted documents properly would be likely to improve the baseline accuracy.

In each test, three cases are considered:

1. We use simple weights, i.e., $W_{ij} = 1$;
2. We use heat kernel weights, i.e., $W_{ij} = exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/t)$
3. We first perform PCA to remove the noise associated with the smallest eigenvalues, then perform LPP with heat kernel. In PCA stage, we kept 98% information in the sense of reconstruction error. Specifically, we first reduce the original space to a k-dimensional space by PCA such that

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_i \lambda_i} \approx 98\%$$

   where λ_i are the eigenvalues of PCA.

In all the tests, we compare the performance of the LPP algorithm with that of Latent Semantic Indexing (LSI), a common technique for text classification and retrieval [20]. LSI preserves (to the extent possible) the relative distances in the term-document matrix while projecting it to a lower-dimensional space. In contrast, LPP preserves local information which directly relates to the nearest neighbor search. The results are summarized in Table 2.1, 2.2, 2.3. LPP outperforms LSI in all three tests. PCA+LPP performs the best when the dimensionality is very low.

It is worthwhile to point out that, in these experiments, we didn't use any label information in the training data. However, the label information can be easily incorporated into the LPP algorithm while constructing the nearest neighbor graph. For example, for each training document, we can restrict to find its k nearest neighbors in the same class. In this way, LPP works in a supervised manner. We believe that better performance can be obtained when label information is incorporated.

CHAPTER 3
KERNEL LPP

3.1 Derivation of LPP in Reproducing Kernel Hilbert Space

In this section, we generalize LPP to nonlinear problems and develop Kernel Locality Preserving Projections (KLPP). KLPP combines the advantages of both Laplacian Eigenmaps and LPP in that it can discover the nonlinear structure of the data manifold as Laplacian Eigenmaps and also produce a mapping function from the input space to the low-dimensional representation space as LPP.

We seek a *minimum norm* function $f \in \mathcal{H}_K$ such that the following objective function is minimized,

$$\min_{f \in \mathcal{H}_K} \sum_{i=1}^{n} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 W_{ij} \tag{3.1}$$

Proposition 1. *Let $\mathcal{H} = \{\sum_{i=1}^{n} \alpha_i K(\cdot, \mathbf{x}_i) | \alpha_i \in \mathbf{R}\}$ be a subspace of $\mathcal{H}_K$, the* minimum norm *solution to the problem (3.1) is in $\mathcal{H}$.*

Proof. Let $\mathcal{H}^{\perp}$ be the orthogonal complement of $\mathcal{H}$, i.e. $\mathcal{H}_K = \mathcal{H} \oplus \mathcal{H}^{\perp}$. Thus, for any function $f \in \mathcal{H}_K$, it has orthogonal decomposition as follows:

$$f = f_{\mathcal{H}} + f_{\mathcal{H}^{\perp}}$$

Now, let's evaluate f at $\mathbf{x}_i$:

$$\begin{aligned} f(\mathbf{x}_i) &= \langle f, K_{\mathbf{x}_i} \rangle_{\mathcal{H}_K} && (3.2) \\ &= \langle f_{\mathcal{H}} + f_{\mathcal{H}_{\perp}}, K_{\mathbf{x}_i} \rangle_{\mathcal{H}_K} && (3.3) \\ &= \langle f_{\mathcal{H}}, K_{\mathbf{x}_i} \rangle_{\mathcal{H}_K} + \langle f_{\mathcal{H}^{\perp}}, K_{\mathbf{x}_i} \rangle_{\mathcal{H}_K} && (3.4) \end{aligned}$$

Notice that $K_{\mathbf{x}_i} \in \mathcal{H}$ while $f_{\mathcal{H}^{\perp}} \in \mathcal{H}^{\perp}$. This implies that $\langle f_{\mathcal{H}^{\perp}}, K_{\mathbf{x}_i} \rangle_{\mathcal{H}_K} = 0$. Therefore,

$$f(\mathbf{x}_i) = \langle f_{\mathcal{H}}, K_{\mathbf{x}_i} \rangle_{\mathcal{H}_K} = f_{\mathcal{H}}(\mathbf{x}_i)$$

This completes the proof. □

Since the solutions to the problem (3.1) are in $\mathcal{H}$, we use $\mathcal{H}_K$ and $\mathcal{H}$ interchangeably thereafter. Thus, the inner product between $f, g \in \mathcal{H}$ where $f(\cdot) = \sum_{i=1}^{n} \alpha_i K(\cdot, \mathbf{x}_i)$ and $g(\cdot) = \sum_{j=1}^{n} \beta_j K(\cdot, \mathbf{x}_j)$ is $\langle f, g \rangle = \sum_{i,j} \alpha_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j)$. If the kernel function is chosen as inner product $K(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$, then $\mathcal{H}_K$ is a linear functional space and the algorithm reduces to ordinary LPP. For general kernel function K and $f \in \mathcal{H}_K$, we have

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i) = K(\mathbf{x})^T \alpha \tag{3.5}$$

where $\alpha = [\alpha_1, \cdots, \alpha_n]^T$ and $K(\mathbf{x}) \doteq [K(\mathbf{x}, \mathbf{x}_1), \cdots, K(\mathbf{x}, \mathbf{x}_n)]^T$. We define

$$\begin{aligned}
\mathbf{y} &= (f(\mathbf{x}_1), \cdots, f(\mathbf{x}_n))^T && (3.6)\\
&= \begin{pmatrix} K(\mathbf{x}_1)^T \alpha \\ \vdots \\ K(\mathbf{x}_1)^T \alpha \end{pmatrix} && (3.7)\\
&= \begin{pmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & \cdots & K(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & \cdots & K(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} \alpha && (3.8)\\
&\doteq K\alpha && (3.9)
\end{aligned}$$

where K is the kernel matrix, $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. Note that, K is symmetric. Thus, the objective function can be reduced to

$$\begin{aligned}
& \sum_{i=1}^{n} (f(\mathbf{x}_i - f(\mathbf{x})_j))^2 W_{ij} && (3.10)\\
=\ & 2\sum_{i=1}^{n} f(\mathbf{x}_i) D_{ii} f(\mathbf{x}_i) - 2\sum_{i=1}^{n} f(\mathbf{x}_i) W_{ij} f(\mathbf{x}_j) && (3.11)\\
=\ & 2\mathbf{y}^T D \mathbf{y} - 2\mathbf{y}^T W \mathbf{y} && (3.12)\\
=\ & 2\mathbf{y}^T L \mathbf{y} && (3.13)\\
=\ & 2\alpha^T K L K \alpha && (3.14)
\end{aligned}$$

where L is the graph Laplacian. Similarly, the constraint can be derived as follows:

$$\mathbf{y}^T D\mathbf{y} = 1 \Rightarrow \alpha^T KDK\alpha = 1 \tag{3.15}$$

Therefore, the optimal mapping function $f \in \mathcal{H}_K$ can be obtained by solving the following minimization problem

$$\begin{array}{l} \arg \min\limits_{\alpha} \quad \alpha^T KLK\alpha \\ \alpha^T KDK\alpha = 1 \end{array} \tag{3.16}$$

This leads to the following generalized eigenvector problem:

$$KLK\alpha = \lambda KDK\alpha \tag{3.17}$$

To get nonlinear LPP, we simply choose a nonlinear kernel. Also, it is important to note that there is no nonlinear optimization involved in kernel LPP, hence it can be computed as simply as the standard LPP.

3.2 Derivation of Locally Linear Embedding Algorithm in Reproducing Kernel Hilbert Space

In this section, we present the derivation of LLE in RKHS which will lead to the linear version of LLE and Kernel LLE.

Given a data set LLE first discovers the adjacency information of the data set. For each data point, we find its k nearest neighbors. The local geometry of the neighborhoods can be characterized by linear coefficients that reconstruct each data point from its neighbors. Reconstruction errors are measured by the cost function [61]:

$$\epsilon(W) = \sum_i |\mathbf{x}_i - \sum_j W_{ij}\mathbf{x}_j|^2 \tag{3.18}$$

which adds up the squared distances between all the data points and their reconstructions. Consider the problem of mapping the original data points to a line so that each data point in the reduced space can be represented as a linear combination of its neighbors with the coefficients W_{ij}. Let $\mathbf{y} = (y_1, y_2, \cdots, y_n)^T$ be such a map. A reasonable criterion for choosing a "good" map is to minimize the following cost function [61]:

$$\Phi(\mathbf{y}) = \sum_i |y_i - \sum_j W_{ij} y_j|^2 \tag{3.19}$$

under appropriate constraints. This cost function, like the previous one, is based on locally linear reconstruction errors, but here we fix the weights W_{ij} while optimizing the coordinates y_i. The optimal y_i can be obtained by solving a eigenvalue problem [61]. However, LLE is defined only on the training data points.

Similar to the analysis presented in previous subsection, LLE can also be solved in RKHS. Let f be a mapping function. $y_i = f(\mathbf{x}_i), f \in \mathcal{H}_K$. The optimization problem in RKHS is as follows,

$$\Phi(f) = \sum_i (f(\mathbf{x}_i) - \sum_j W_{ij} f(\mathbf{x}_j))^2 \tag{3.20}$$

We define

$$z_i = f(\mathbf{x}_i) - \sum_j W_{ij} f(\mathbf{x}_j) \tag{3.21}$$

Let $\mathbf{y} = [f(\mathbf{x}_1, \cdots, f(\mathbf{x}_n)]^T = K\alpha$ and $\mathbf{z} = [z_1, \cdots, z_n]^T$. We have

$$\begin{aligned} \mathbf{z} &= \mathbf{y} - W\mathbf{y} & (3.22)\\ &= (I - W)\mathbf{y} & (3.23)\\ &= (I - W)K\alpha & (3.24) \end{aligned}$$

Thus, the cost function can be reduced to

$$\begin{aligned}\Phi(f) &= \sum_i (f(\mathbf{x}_i) - \sum_j W_{ij} f(\mathbf{x}_j))^2 && (3.25)\\ &= \sum_i (z_i)^2 && (3.26)\\ &= \mathbf{z}^T\mathbf{z} && (3.27)\\ &= \alpha^T K(I-W)^T(I-W)K\alpha && (3.28)\\ &= \alpha^T KMK\alpha && (3.29)\end{aligned}$$

where $M = (I-W)^T(I-W)$. In order to remove an arbitrary scaling factor in the mapping, we impose a constraint as follows:

$$\|\mathbf{y}\|^2 = 1 \Rightarrow (K\alpha)^T(K\alpha) = \alpha^T K^2 \alpha = 1 \tag{3.30}$$

Thus, we get the following minimization problem:

$$\underset{\substack{\alpha \\ \alpha^T K^2 \alpha = 1}}{\arg\ \min} \quad \alpha^T KMK\alpha \tag{3.31}$$

which leads to the following generalized eigenvalue problem:

$$KMK\alpha = K^2\alpha \tag{3.32}$$

Thus, the optimal mapping function is $f = \sum_i \alpha_i K(\cdot, \mathbf{x}_i)$ which is defined everywhere.

Finally, we consider the special case that f is linear, i.e. $y_i = f(\mathbf{x}_i) = \mathbf{x}_i^T\mathbf{a}$. Define $X = [\mathbf{x}_1, \cdots, \mathbf{x}_n]^T$. Thus, $\mathbf{z} = (I-W)X^T\mathbf{a}$. The cost function of $\mathbf{a}$ is

$$\phi(\mathbf{a}) = \mathbf{z}^T\mathbf{z} = \mathbf{a}^T X(I-W)^T(I-W)X^T\mathbf{a} = \mathbf{a}^T XMX^T\mathbf{a} \tag{3.33}$$

The constraint is

$$\mathbf{y}^T\mathbf{y} = 1 \Rightarrow \mathbf{a}^T XX^T\mathbf{a} = 1 \tag{3.34}$$

We get the following minimization problem when f is linear:

$$\underset{\substack{\mathbf{a} \\ \mathbf{a}^T X X^T \mathbf{a} = 1}}{\arg \min} \quad \mathbf{a}^T X M X^T \mathbf{a} \tag{3.35}$$

which leads to the following eigenvalue problem:

$$X M X^T \mathbf{a} = X X^T \mathbf{a} \tag{3.36}$$

Finally, it is important to note that our derivation of LLE in RKHS yield a map (either linear or nonlinear depending on the chosen reproducing kernel) which is defined everywhere while the original LLE is defined only on the training data points. Some other work on out-of-sample extensions can be found in [11].

CHAPTER 4
FACE ANALYSIS USING LAPLACIANFACES

In this section, we introduce a new appearance based face recognition method, called *Laplacianfaces*. The Laplacianface method is fundamentally based on LPP.

4.1 Appearance Based Face Analysis

Many face recognition techniques have been developed over the past few decades. One of the most successful and well-studied techniques to face recognition is the appearance-based method [55] [73]. When using appearance-based methods, we usually represent an image of size n m pixels by a vector in an n m dimensional space. In practice, however, these n m dimensional spaces are too large to allow ro-bust and fast face recognition. A common way to attempt to resolve this problem is to use dimensionality reduction techniques [7] [9] [29] [48] [52] [66] [70] [73] [83] [87]. Two of the most popular techniques for this purpose are Principal Component Analysis (PCA) [40] and Linear Discriminant Analysis (LDA) [22].

PCA is an eigenvector method designed to model linear variation in high-dimensional data. PCA performs dimensionality reduction by projecting the original n-dimensional data onto the k (¡¡n)-dimensional linear subspace spanned by the leading eigenvectors of the data's covariance matrix. Its goal is to find a set of mutually orthogonal basis functions that capture the directions of maximum variance in the data and for which the coefficients are pairwise de-correlated. For linearly embedded manifolds, PCA is guaranteed to discover the dimensionality of the manifold and produces a compact representation. Turk and Pentland [73] use Principal Component Analysis to describe face images in terms of a set of basis functions, or "Eigenfaces".

LDA is a supervised learning algorithm. LDA searches for the project axes on which the data points of different classes are far from each other while requiring data points of the same class to be close to each other. Unlike PCA which encodes information in an orthogonal linear space, LDA encodes discriminating information in a linearly separable space using bases that are not necessarily orthogonal. It is generally believed that algorithms based on LDA are superior to those based on PCA. However, some recent work [52] shows that, when the training dataset is small, PCA can outperform LDA, and also that PCA is less sensitive to different training datasets.

Recently, a number of research efforts have shown that the face images possibly reside on a nonlinear sub-manifold [16] [47] [61] [62][65] [2] [71]. However, both PCA and LDA effectively see only the Euclidean structure. They fail to discover the underlying structure, if the face images lie on a nonlinear sub-manifold hidden in the image space. Some nonlinear techniques have been proposed to discover the nonlinear structure of the manifold, e.g. Isomap [71], LLE [61][62], and Laplacian Eigenmaps [10]. These nonlinear methods do yield impressive results on some benchmark artificial data sets. However, they yield maps that are defined only on the training data points and how to evaluate the maps on novel test data points remains unclear. Therefore, these nonlinear manifold learning techniques [10][15][61] [63] [71][86] might not be suitable for some computer vision tasks, such as face recognition.

In the meantime, there has been some interest in the problem of developing low dimensional representations through kernel based techniques for face recognition [50] [85]. These methods can discover the nonlinear structure of the face images. However, they are computationally expensive. Moreover, none of them explicitly considers the structure of the manifold on which the face images possibly reside.

In this paper, we propose a new approach to face analysis (representation and recognition), which explicitly considers the manifold structure. To be specific, the manifold structure is modelled by a nearest-neighbor graph which preserves the local structure of the image space. A face subspace is obtained by Locality Preserving Projections (LPP) [33]. Each face image in the image space is mapped to a low-dimensional face subspace, which is characterized by a set of feature images, called **Laplacianfaces**. The face subspace preserves local structure, and seems to have more discriminating power than the PCA approach for classification purpose. We also provide theoretical analysis to show that PCA, LDA and LPP can be obtained from different graph models. Central to this is a graph structure that is inferred on the data points. LPP finds a projection that respects this graph structure. In our theoretical analysis, we show how PCA, LDA, and LPP arise from the same principle applied to different choices of this graph structure.

It is worthwhile to highlight several aspects of the proposed approach here:

1. While the Eigenfaces method aims to preserve the global structure of the image space, and the Fisher-faces method aims to preserve the discriminating information; our Laplacianfaces method aims to pre-serve the local structure of the image space. In many real world classification problems, the local manifold structure is more important than the global Euclidean structure, especially when nearest-neighbor like classifiers are used for classification. LPP seems to have discriminating power although it is unsupervised.

2. An efficient subspace learning algorithm for face recognition should be able to discover the nonlinear manifold structure of the face space. Our proposed Laplacianfaces method explicitly considers the manifold structure which is modelled by an adjacency graph. Moreover, the Laplacianfaces are obtained by finding the optimal linear approximations to the eigenfunctions of the Laplace Beltrami operator on the face manifold. They reflect the intrinsic face manifold structures.

3. LPP shares some similar properties to LLE [61], such as a locality preserving character. However, their objective functions are totally different. LPP is obtained by finding the optimal linear approximations to the eigenfunctions of the Laplace Beltrami operator on the manifold. LPP is linear, while LLE is nonlinear. Moreover, LPP is defined everywhere, while LLE is defined only on the training data points and it is unclear how to evaluate the maps for new test points. In contrast, LPP may be simply applied to any new data point to locate it in the reduced representation space.

4.2 Manifold Ways for Visual Analysis

In the above sections, we have described three different linear subspace learning algorithms. The key difference between PCA, LDA and LPP is that, PCA and LDA aim to discover the global structure of the Euclidean space, while LPP aims to discover local structure of the manifold. In this section, we discuss the manifold ways of visual analysis.

4.2.1 Manifold Learning via Dimensionality Reduction

In many cases, face images may be visualized as points drawn on a low-dimensional manifold hid-den in a high-dimensional ambient space. Specially, we can consider that a sheet of rubber is crumpled into a (high-dimensional) ball. The objective of a dimensionality-reducing mapping is to unfold the sheet and to make its low-dimensional structure explicit. If the sheet is not torn in the process, the mapping is topology-preserving. Moreover, if the rubber is not stretched or compressed, the mapping preserves the metric structure of the original space. In this paper, our objective is to discover the face manifold by a locally topology-preserving mapping for face analysis (representation and recognition).

4.2.2 Learning Laplacianfaces for Representation

LPP is a general method for manifold learning. It is obtained by finding the optimal linear approximations to the eigenfunctions of the Laplace Betrami operator on the manifold [33]. Therefore, though it is still a linear technique, it seems to recover important aspects of the intrinsic nonlinear manifold structure by preserving local structure. Based on LPP, we describe our Laplacianfaces method for face representation in a locality preserving subspace.

In the face analysis and recognition problem, one is confronted with the difficulty that the matrix XDX^T is sometimes singular. This stems from the fact that sometimes the number of images in the training set (n) is much smaller than the number of pixels in each image (m). In such a case, the rank of XDX^T is at most n, while XDX^T is an m m matrix, which implies that XDX^T is singular. To overcome the complication of a singular XDX^T, we first project the image set to a PCA subspace so that the resulting matrix XDX^T is nonsingular. Another consideration of using PCA as preprocessing is for noise reduction. This method, we call Laplacianfaces, can learn an optimal subspace for face representation and recognition.

The algorithmic procedure of Laplacianfaces is formally stated below:

1. **PCA Projection**: We project the image set $\{\mathbf{x}_i\}$ into the PCA subspace by throwing away the smallest principle components. In our experiments, we kept 98% information in the sense of reconstruction error. For the sake of simplicity, we still use $\mathbf{x}$ to denote the images in the PCA subspace in the following steps. We denote by A_{PCA} the transformation matrix of PCA.

2. **Constructing the nearest-neighbor graph**: Let G denote a graph with n nodes. The i^{th} node corresponds to the face image $\mathbf{x}_i$. We put an edge between nodes i and j if $\mathbf{x}_i$ and $\mathbf{x}_j$ are "close", i.e. $\mathbf{x}_i$ is among k nearest neighbors of $\mathbf{x}_j$ or $\mathbf{x}_j$ is among k nearest neighbors of $\mathbf{x}_i$. The constructed nearest-neighbor graph is an approximation of the local manifold structure. Note that, here we do not use the ϵ-neighborhood to construct the graph. This is simply because it is often difficult to choose the optimal ϵ in the real world applications, while k nearest neighbor graph can be constructed more stably. The disadvantage is that the k nearest neighbor search will increase the computational complexity of our algorithm. When the computational complexity is a major concern, one can switch to the ϵ-neighborhood.

3. **Choosing the weights**: If node i and j are connected, put

$$S_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{t}} \tag{4.1}$$

where t is a suitable constant. Otherwise, put $S_{ij} = 0$. The weight matrix S of graph G models the face manifold structure by preserving local structure. The justification for this choice of weights can be traced back to [10].

4. **Eigenmaps**: Compute the eigenvectors and eigenvalues for the generalized eigenvector problem:

$$XLX^T\mathbf{a} = \lambda XDX^T\mathbf{a} \tag{4.2}$$

where D is a diagonal matrix whose entries are column (or row, since S is symmetric) sums of S, $D_{ii} = \Sigma_j S_{ji}$. $L = D - S$ is the Laplacian matrix. The i^{th} column of matrix X is $\mathbf{x}_i$.

Let the column vectors $\mathbf{a}_0, \mathbf{a}_1, \cdots, \mathbf{a}_{l-1}$ be the solutions of equation (4.2), ordered according to their eigenvalues, $\lambda_0 < \lambda_1 < \cdots < \lambda_{l-1}$. Thus, the embedding is as follows:

$$\mathbf{x}_i \rightarrow \mathbf{y}_i = A^T \mathbf{x}_i \tag{4.3}$$

$$A = A_{PCA} A_{LPP} \tag{4.4}$$

$$A_{LPP} = \begin{pmatrix} \mathbf{a}_0, & \mathbf{a}_1, & \cdots, & \mathbf{a}_{l-1} \end{pmatrix} \tag{4.5}$$

where $\mathbf{y}_i$ is a l-dimensional vector, and A is a $n \times l$ matrix.

4.2.3 Face Manifold Analysis

Now consider a simple example of image variability. Imagine that a set of face images are generated while the human face rotates slowly. Intuitively, the set of face images correspond to a continuous curve in image space, since there is only one degree of freedom, viz. the angel of rotation. Thus, we can say that the set of face images are intrinsically one-dimensional. Many recent works [16] [47] [61][62][65] [2] [71] have shown that the face images do reside on a low-dimensional sub-manifold embedded in a high-dimensional ambient space (image space). Therefore, an effective subspace learning algorithm should be able to detect the nonlinear manifold structure. The conventional algorithms, such as PCA and LDA, model the face images in Euclidean space. They effectively see only the Euclidean structure; thus, they fail to detect the intrinsic low-dimensionality.

With its neighborhood preserving character, the Laplacianfaces seem to be able to capture the intrinsic face manifold structure to a larger extent. Figure (4.1) shows an example that the face images with various pose and expression of a person are mapped into two-dimensional subspace. The face image dataset used here is the same as that used in [61]. This dataset contains 1965 face images taken from se-quential frames of a small video. The size of each image is 20×28 pixels, with 256 gray levels per pixel.

Thus, each face image is represented by a point in the 560-dimensional ambient space. However, these images are believed to come from a sub-manifold with few degrees of freedom. We leave out 10 samples for testing, and the remaining 1955 samples are used to learn the Laplacianfaces. As can be seen, the face images are mapped into a two-dimensional space with continuous change in pose and expression. The representative face images are shown in the different parts of the space. The face images are divided into two parts. The left part includes the face images with open mouth, and the right part includes the face images with closed mouth. This is because in trying to preserve local structure in the embedding, the Laplacianfaces implicitly emphasizes the natural clusters in the data. Specifically, it makes the neighboring points in the image face nearer in the face space, and faraway points in the image face farther in the face space. The bottom images correspond to points along the right path (linked by solid line), illustrating one particular mode of variability in pose.

The ten testing samples can be simply located in the reduced representation space by the Laplacianfaces (column vectors of the matrix W). Figure (4.2) shows the result. As can be seen, these testing samples optimally find their coordinates which reflect their intrinsic properties, i.e. pose and expression. This observation tells us that the Laplacianfaces are capable of capturing the intrinsic face manifold structure to some extent.

Recall that both Laplacian Eigenmaps and LPP aim to find a map which preserves the local structure. Their objective function is as follows:

$$\min_f \sum_{ij} \left(f(\mathbf{x}_i) - f(\mathbf{x}_j)\right)^2 S_{ij} \tag{4.6}$$

The only difference between them is that, LPP is linear while Laplacian Eigenmap is non-linear. Since they have the same objective function, it would be important to see to what extent LPP can approximate Laplacian Eigenmap. This can be evaluated by comparing their eigenvalues. Figure (4.3) shows the eigenvalues computed by the two methods. As can be seen, the eigenvalues of LPP is consistently greater than those of Laplacian Eigenmaps, but the difference between them is small. The difference gives a measure of the degree of the approximation.

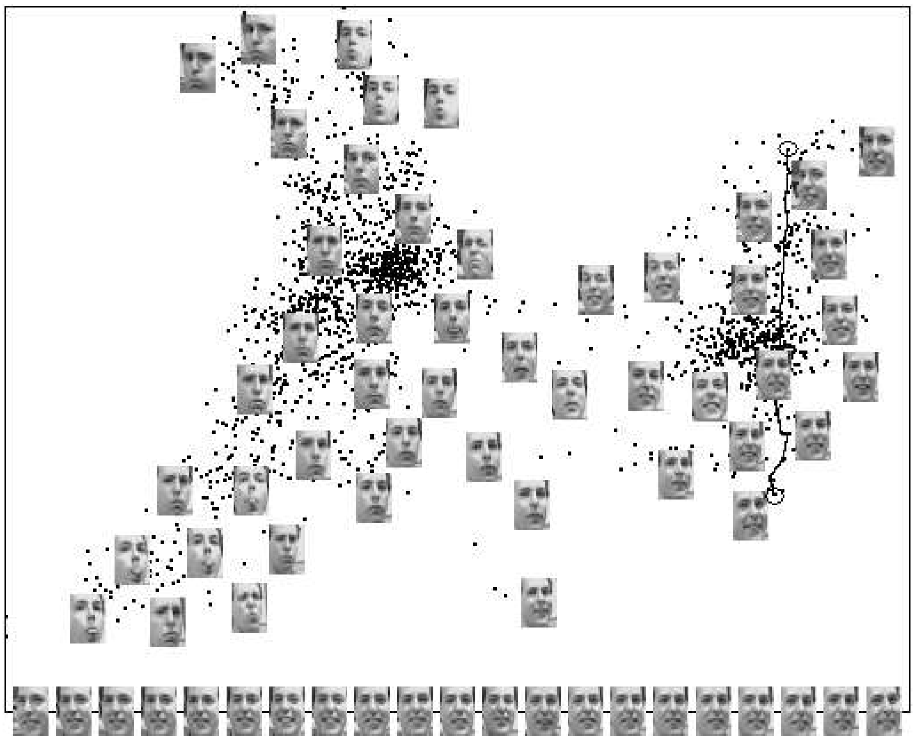


Figure 4.1: Two-dimensional linear embedding of face images by Laplacianfaces. As can be seen, the face images are divided into two parts, the faces with open mouth and the faces with closed mouth. Moreover, it can be clearly seen that the pose and expression of human faces change continuously and smoothly, from top to bottom, from left to right. The bottom images correspond to points along the right path (linked by solid line), illustrating one particular mode of variability in pose.

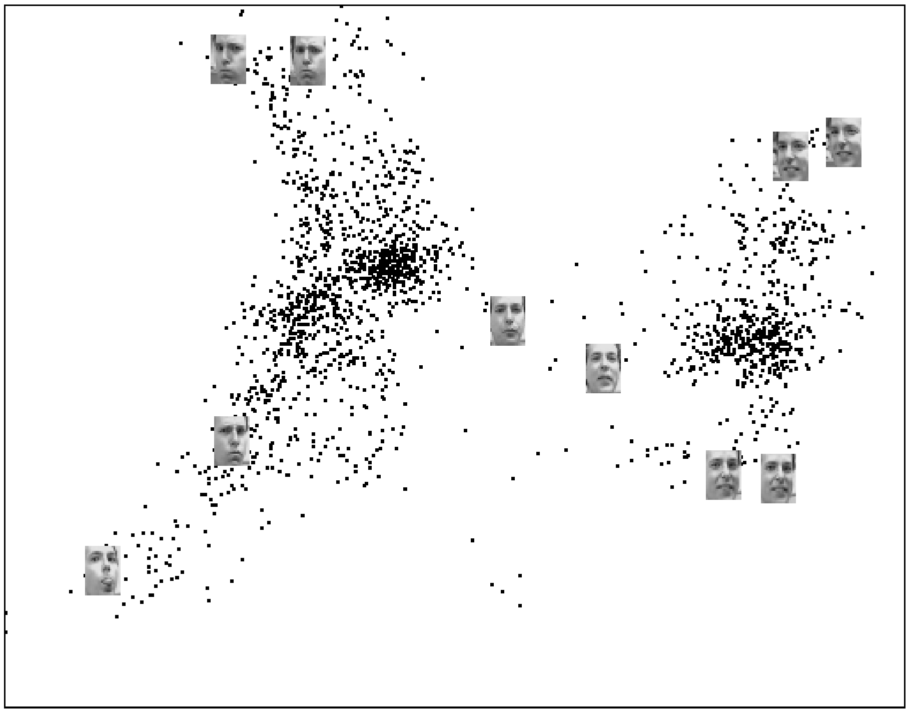


Figure 4.2: Distribution of the 10 testing samples in the reduced representation subspace. As can be seen, these testing samples optimally find their coordinates which reflect their intrinsic properties, i.e. pose and expression.

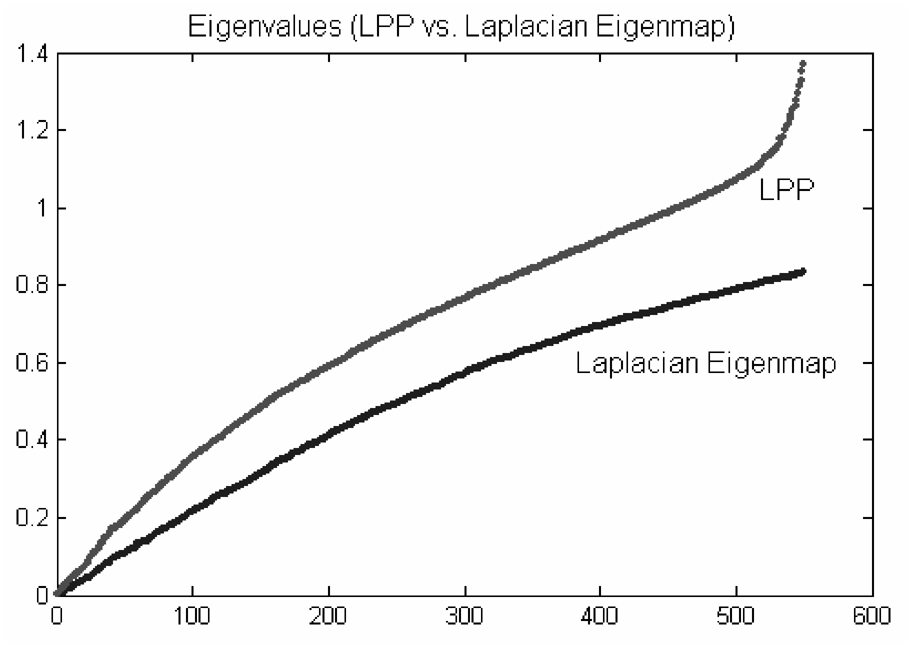


Figure 4.3: The eigenvalues of LPP and Laplacian Eigenmap.

4.3 Experimental Results

In Section 2.7, we have presented some simple synthetic examples to show that LPP can have more discriminative power than PCA and be less sensitive to outliers. In this section, several experiments are carried out to show the effectiveness of our proposed Laplacianface method for face representation and recognition.

4.3.1 Face Representation Using Laplacianfaces

As we described previously, a face image can be represented as a point in image space. A typical image of size $m \times n$ describes a point in $m \times n$-dimensional image space. However, due to the unwanted variations resulting from changes in lighting, facial expression, and pose, the image space might not be an optimal space for visual representation.

In Section 5.2.2, we have discussed how to learn a locality preserving face subspace which is insensitive to outlier and noise. The images of faces in the training set are

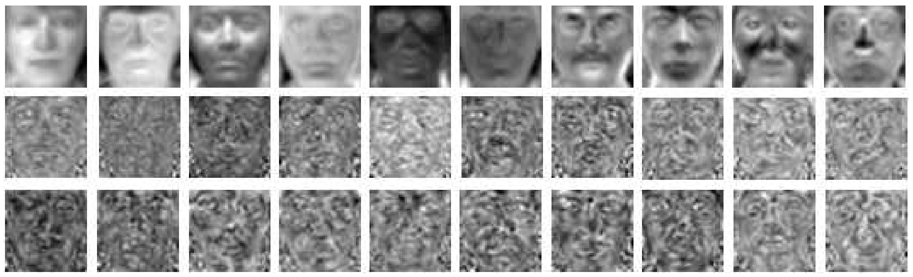


Figure 4.4: The first 10 Eigenfaces (first row), Fisherfaces (second row) and Laplacianfaces (third row) calculated from the face images in the YALE database.

used to learn such a locality preserving subspace. The subspace is spanned by a set of eigenvectors of equation (4.2). We can display the eigenvectors as images. These images may be called Laplacianfaces. Using the Yale face database as the training set, we present the first 10 Laplacianfaces in Figure (4.4), together with Eigenfaces and Fisherfaces. A face image can be mapped into the locality preserving subspace by using the Laplacianfaces. It is interesting to note that the Laplacianfaces are somehow similar to Fisherfaces.

4.3.2 Face Recognition Using Laplacianfaces

Once the Laplacianfaces are created, face recognition [9][52][72] [79] becomes a pattern classification task. In this section, we investigate the performance of our proposed Laplacianfaces method for face recognition. The system performance is compared with the Eigenfaces method [72] and the Fisherfaces method [9], two of the most popular linear methods in face recognition.

In this study, three face databases were tested. The first one is the PIE (pose, illumination, and expression) database from CMU [69], the second one is the Yale database [1], and the third one is the MSRA database collected at the Microsoft Research Asia. In all the experiments, preprocessing to locate the faces was applied. Original images were normalized (in scale and orientation) such that the two eyes were aligned at the same position. Then, the facial areas were cropped into the final images for matching. The size of each cropped image in all the experiments

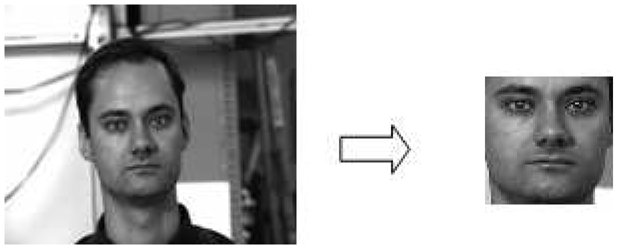


Figure 4.5: The original face image and the cropped image.

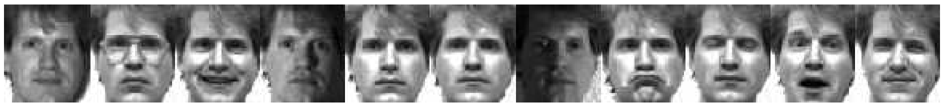


Figure 4.6: Sample face images of an individual from Yale face database.

is 32×32 pixels, with 256 grey levels per pixel. Thus, each image is represented by a 1024-dimensional vector in image space. The details of our methods for face detection and alignment can be found in [80] [82]. No further preprocessing is done. Figure (4.5) shows an example of the original face image and the cropped image. Different pattern classifiers have been applied for face recognition, including nearest-neighbor [9], Bayesian [53], Support Vector Machine [58], etc. In this paper, we apply the nearest-neighbor classifier for its simplicity. The Euclidean metric is used as our distance measure. However, there might be some more sophisticated and better distance metric, e.g. variance-normalized distance, which may be used to improve the recognition performance. The number of nearest neighbors in our algorithm was set to be 4 or 7, according to the size of the training set.

In short, the recognition process has three steps. First, we calculate the Laplacianfaces from the training set of face images; then the new face image to be identified is projected into the face subspace spanned by the Laplacianfaces; finally, the new face image is identified by a nearest-neighbor classifier.

Table 4.1: Performance comparison on the Yale database

Approach	Dimension	Error rate
Eigenface	33	25.3%
Fisherface	14	20.0%
Laplacianface	28	11.3%

YALE DATABASE

The Yale face database [1] was constructed at the Yale Center for Computational Vision and Control. It contains 165 grayscale images of 15 individuals. The images demonstrate variations in lighting condition (left-light, center-light, right-light), facial expression (normal, happy, sad, sleepy, surprised, and wink), and with/without glasses. Figure 4.6 shows some sample face images of one individual.

A random subset with six images per individual (hence 90 images in total) was taken with labels to form the training set. The rest of the database was considered to be the testing set. The training samples were used to learn the Laplacianfaces. The testing samples were then projected into the low-dimensional representation subspace. Recognition was performed using a nearest neighbor classifier.

Note that, for LDA, there are at most $c-1$ nonzero generalized eigenvalues, and so an upper bound on the dimension of the reduced space is $c-1$, where c is the number of individuals [9]. In general, the performance of the Eigenfaces method and the Laplacianfaces method varies with the number of dimensions. We show the best results obtained by Fisherfaces, Eigenfaces, and Laplacianfaces.

The recognition results are shown in Table 4.3.2. It is found that the Laplacianfaces method significantly outperforms both Eigenfaces and Fisherfaces methods. The recognition approaches the best results with 14, 28, 33 dimensions for Fisherfaces, Laplacianfaces, and Eigenfaces respectively. The error rates of Fisherfaces, Laplacianfaces, and Eigenfaces are 20%, 11.3%, and 25.3% respectively. The correspond-ing face subspaces are called optimal face subspace for each method. There is no significant improvement if more dimensions are used. Figure 4.7 shows the plots of error rate vs. dimensionality reduction.

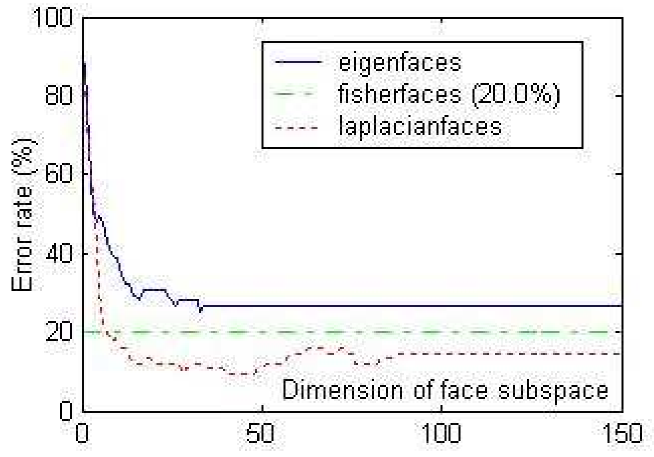


Figure 4.7: Recognition accuracy vs. dimensionality reduction on Yale database.

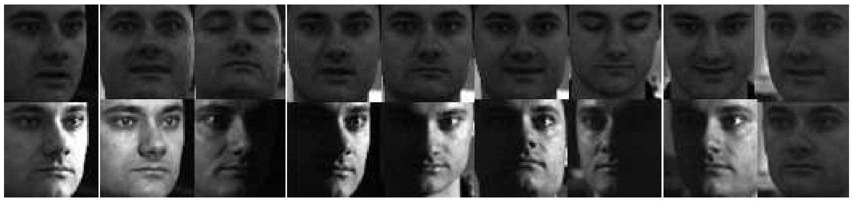


Figure 4.8: The sample cropped face images of one individual from PIE database. The original face images are taken under varying pose, illumination, and expression.

PIE DATABASE

The CMU PIE face database contains 68 subjects with 41,368 face images as a whole. The face images were captured by 13 synchronized cameras and 21 flashes, under varying pose, illumination and expression. We used 170 face images for each individual in our experiment, 85 for training and the other 85 for testing. Figure 4.8 shows some of the faces with pose, illumination and expression variations in the PIE database.

Table 4.3.2 shows the recognition results. As can be seen, Fisherfaces performs comparably to our algorithm on this database, while Eigenfaces performs poorly. The error rate for Laplacianfaces, Fisher-faces, and Eigenfaces are 4.6%, 5.7%, and 20.6%, respectively. Figure 4.9 shows a plot of error rate vs. dimensionality reduction. As can be seen, the error rate of our Laplacianfaces method decreases fast as the di-

Table 4.2: Performance comparison on the PIE database

Approach	Dimensi on	Error rate
Eigenface	150	20.6%
Fisherface	67	5.7%
Laplacianface	110	4.6%

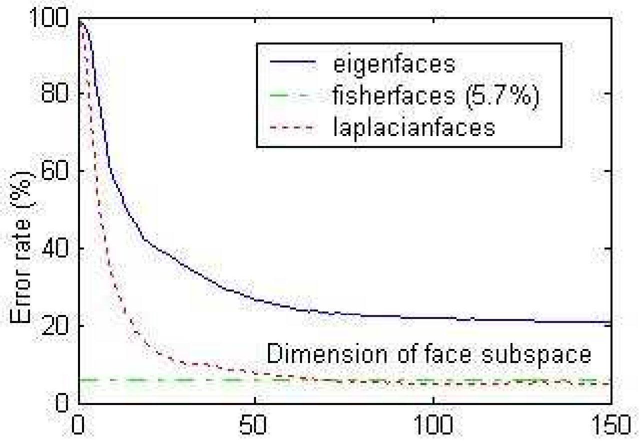


Figure 4.9: Recognition accuracy vs. dimensionality reduction on PIE database.

Table 4.3: Performance comparison on the MSRA database

Approach	Dimensi on	Error rate
Eigenface	142	35.4%
Fisherface	11	26.5%
Laplacianface	66	8.2%

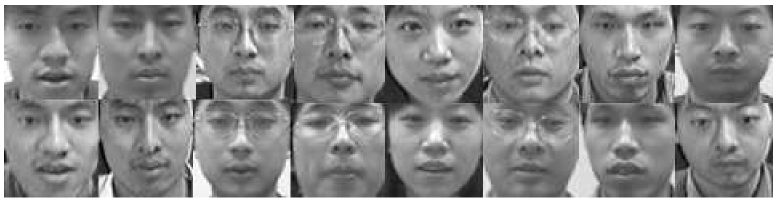


Figure 4.10: The sample cropped face images of 8 individuals from MSRA database. The face images in the first row are taken in the first session, which are used for training. The face images in the second row are taken in the second session, which are used for testing. The two images in the same column are corresponding to the same individual.

mensionality of the face subspace increases, and achieves the best result with 110 dimensions. There is no significant improvement if more dimensions are used. Eigenfaces achieves the best result with 150 dimensions. For Fisherfaces, the dimension of the face subspace is bounded by $c - 1$, and it achieves the best result with $c - 1$ dimensions. The dashed horizontal line in the figure shows the best result obtained by Fisherfaces.

MSRA DATABASE

This database was collected at Microsoft Research Asia. It contains 12 individuals, captured in two different sessions with different backgrounds and illuminations. 64 to 80 face images were collected for each individual in each session. All the faces are frontal. Figure 4.10 shows the sample cropped face images from this database. In this test, one session was used for training and the other was used for testing. Table 4.3.2 shows the recognition results. Laplacianfaces method has lower error rate (8.2%) than those of Eigenfaces (35.4%) and Fisherfaces (26.5%). Figure 4.11 shows

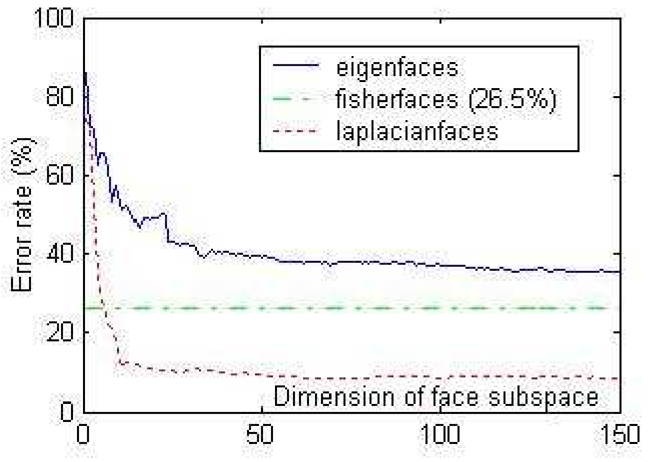


Figure 4.11: Recognition accuracy vs. dimensionality reduction on MSRA database.

Table 4.4: Recognition error rate with different number of training samples

	Percentage of training samples in MSRA-S1			
	10%	40%	70%	100%
Eigenface	46.5%	37.8%	38.4%	35.4%
Fisherface	27.0%	20.1%	29.4%	26.4%
Laplacianface	26.0%	14.7%	13.2%	8.2%

a plot of error rate vs. dimensionality reduction.

IMPROVEMENT WITH THE NUMBER OF TRAINING SAMPLES

In the above subsections, we have shown that our Laplacianfaces approach outperforms the Eigenfaces and Fisherfaces approaches on three different face databases. In this section, we evaluate the performances of these three approaches with different numbers of training samples. We expect that a good algorithm should be able to take advantage of more training samples.

We used the MSRA database for this experiment. Let MSRA-S1 denote the image set taken in the first session, and MSRA-S2 denote the image set taken in the second session. MSRA-S2 was exclusively used for testing. 10%, 40%, 70%, and 100% face images were randomly selected from MSRA-S1 for training. For each of the first

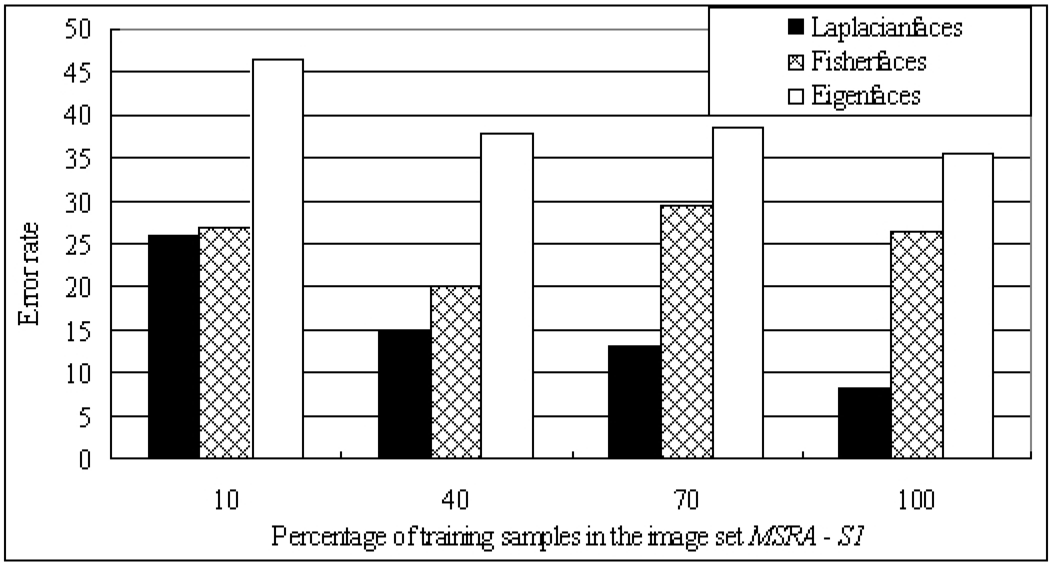


Figure 4.12: Performance comparison on the MSRA database with different number of traning samples. The training samples are from the set MSRA-S1. The images in MSRA-S2 are used for testing exclusively. As can be seen, our Laplacianfaces method takes advantage of more training samples. The error rate reduces from 26.04% with 10% training samples to 8.17% with 100% training samples. There is no evidence for Fisherfaces that it can take advantage of more training samples.

Table 4.5: Performance comparisons on CMU PIE face database

k	Accuracy				Mutual Information			
	Kmeans	PCA	LPP	LE	Kmeans	PCA	LPP	LE
5	0.533	0.580	0.969	0.969	0.514	0.545	0.961	0.961
10	0.469	0.510	0.917	0.917	0.555	0.581	0.951	0.951
20	0.434	0.464	0.824	0.824	0.594	0.607	0.918	0.918
30	0.404	0.426	0.764	0.764	0.599	0.613	0.900	0.900
40	0.393	0.415	0.751	0.751	0.608	0.623	0.900	0.900
50	0.390	0.392	0.729	0.729	0.615	0.624	0.900	0.900
68	0.377	0.387	0.674	0.674	0.635	0.643	0.875	0.875

three cases, we repeated 20 times and computed the error rate in average. In Figure 4.12 and Table 4.3.2 we show the experimental results of the improvement of the performance of our algorithm with the number of training samples. As can be seen, our algorithm takes advantage of more training samples. The performance improves significantly as the number of training sample increases. The error rate of recognition reduces from 26.04% with 10% training samples to 8.17% with 100% training samples. This is due to the fact that, our algorithm aims at discovering the underlying face manifold embedded in the ambient space and more training samples help to recover the manifold structure. Eigenface also takes advantage of more training samples to some extent. The error rate of Eigenface reduces from 46.48% with 10% training samples to 35.42% with 100% training samples. But its performance starts to converge at 40% (training samples) and little improvement is obtained if more training samples are used. For Fisherfaces, there is no convincing evidence that it can take advantage of more training samples.

4.3.3 Face Clustering Using Laplacianfaces

In this Section, we investigate the use of our algorithm on face clustering [26][36][49]. The CMU PIE face database [69] is used in this experiment. The data and evaluation metrics we used in this test is the same as those in Section 6.4.2.

The evaluations were conducted with different number of clusters, ranging from two to ten. For each given number k, k classes were randomly selected from the face database. This process was repeated 20 times (except for $k = 68$) and the average performance was computed. For each single test (given k classes of images), three algorithms, i.e. PCA, LPP and Laplacian Eigenmaps, were used to project the face images into a k dimensional subspace. The K-means was then performed in the subspace as well as the original image space. Again, the K-means was repeated 10 times with different initializations and the best result was recorded. Table 2 shows the experimental results. As can be seen, our algorithm performed much better than K-means and PCA+K-means. Little improvement can be gained by PCA based clustering. This is because that, PCA can extract the most representative features, but it fails to extract the most discriminative features.

4.3.4 Discussion

Three experiments on three databases have been systematically performed. These experiments reveal a number of interesting points:

1. Both Eigenface, Fisherface and Laplacianface performed better in the optimal face subspace than in the original image space.

2. In all the face recognition experiments, Laplacianfaces consistently performs better than Eigenfaces and Fisherfaces. Especially, it significantly outperforms Fisherfaces and Eigenfaces on Yale database and MSRA database. These experiments also show that our algorithm is especially suitable for frontal face images. Moreover, our algorithm takes advantage of more training samples, which is important to the real world face recognition systems.

3. Comparing to the Eigenfaces method, the Laplacianfaces method encodes more discriminating information in the low-dimensional face subspace by preserving local structure which is more important than the global structure for classification, especially when nearest neighbor like classifiers are used. In fact, if there is a reason to believe that Euclidean distances ($\|\mathbf{x}_i - \mathbf{x}_j\|$) are meaningful only if they are small (local), then our algorithm finds a projection that respects such a belief. Another reason is that, as we show in Figure (**??**), the face images

probably reside on a nonlinear manifold. Therefore, an efficient and effective subspace representation of face images should be capable of characterizing the nonlinear manifold structure, while the Laplacianfaces are exactly derived by finding the optimal linear approximations to the eigenfunctions of the Laplace Beltrami operator on the face manifold. By discovering the face manifold structure, Laplacianfaces can identify the person with different expressions, poses, and lighting conditions.

4. In all the face recognition experiments, the number of training samples per subject is greater than one. Sometimes there might be only one training sample available for each subject. In such a case, LDA can not work since the within-class scatter matrix S_W becomes a zero matrix. For LPP, however, we can construct a complete graph and use inner products as its weights. Thus, LPP can give similar result to PCA.

CHAPTER 5
LOCALITY PRESERVING INDEXING FOR DOCUMENT REPRESENTATION

5.1 Document Representation

Document representation and indexing has been a fundamental problem for efficient clustering, classification, and retrieval [3][4][20][81]. A document can be represented as a point in the vector space $\mathbb{R}^n$ (given by the number of terms in the documents) [5]. Throughout this paper, we denote by document space the set of all document vectors. The document space is typically a subspace of $\mathbb{R}^n$, either linear or non-linear. n is typically very large while the intrinsic dimensionality of the document space might be much lower. Mathematically, we can consider the document space as a low dimensional manifold which is embedded in the high dimensional Euclidean space. A document manifold equipped with a Riemannian metric becomes a Riemannian manifold which is a metric space. Thus, we can compute the distances and angels on the document manifold like what we do in the Euclidean space. Many manifold learning and dimensionality reduction techniques [3][4][13][37][38][45] have been applied to discover such a manifold for document representation and indexing. Among these techniques, Latent Semantic Indexing (LSI) [20] by Singular Value Decomposition (SVD) is a well-known successful approach applied to document analysis [23][27] and information retrieval [43].

LSI is essentially a dimensionality reduction technique developed in the context of information retrieval in order to address the problems deriving from the use of synonymous, near-synonymous, and polysemous words as dimensions of document and query representations. Given a term-document matrix X, LSI applies SVD to project the document vectors into a subspace so that cosine similarity can reflect semantic similarity. LSI aims to find the best linear subspace approximation to the original document space in the sense of minimizing the global reconstruction error. In other words, LSI seeks to uncover the most representative features rather the most discriminative features for document representation. Therefore, LSI might not be optimal in discriminating documents with different semantics. Also, when the document space is nonlinear, LSI fails to discover the intrinsic geometrical structure.

Some variants of LSI have been proposed recently, such as probabilistic LSI (PLSI) [37], iterative residual rescaling (IRR) [4], etc. PLSI is based on the likelihood principle, defines a generative data model, and directly minimizes word perplexity. It can also take advantage of statistical standard methods for model fitting, overfitting control, and model combination. IRR is an alternative subspace projection method that outperforms LSI by counteracting its tendency to ignore minority class documents. This is done by repeatedly rescaling vectors to amplify the presence of documents poorly represented in previous iterations. These methods achieved good empirical results on standard databases. However, similar to the standard LSI, these methods effectively see only the global structure of the document space while in many cases the local structure is more important. Moreover, these methods do not take into account the discriminating structure which might be the most important for real world applications.

In this section, we introduce a new approach called Locality Preserving Indexing (LPI) for document representation, which aims to discover the local geometrical structure of the document space. Since the neighboring documents (data points in high dimensional space) probably relate to the same topic, LPI can have more discriminating power than LSI even though LPI is also unsupervised. To be specific, an adjacency graph is constructed to model the local structure of the document space. A semantic space for document representation is learned by using Locality Preserving Projections (LPP) [33] which is a recently proposed algorithm for linear dimensionality reduction. From the perspective of discrimination, we provide a theoretical analysis to show that LPP is an optimal unsupervised approximation to Linear Discriminant Analysis (LDA) [22] based on the assumption that the neighboring documents are probably related to the same topic.

5.2 Locality Preserving Indexing

Given a set of documents $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n \in \mathbb{R}^m$. Suppose $\mathbf{x}_i$ has been normalized to 1, thus the dot product of two document vectors is exactly the cosine similarity of the two documents. The LPI algorithm is performed as follows:

1. **Constructing the adjacency graph**: Let G denote a graph with n nodes. The i-th node corresponds to the document $\mathbf{x}_i$. We put an edge between nodes i and j if $\mathbf{x}_i$ and $\mathbf{x}_j$ are "close", i.e. $\mathbf{x}_i$ is among p nearest neighbors of $\mathbf{x}_j$ or $\mathbf{x}_j$ is among p nearest neighbors of $\mathbf{x}_i$.

2. **Choosing the weights**: If node i and j are connected, put

$$S_{ij} = \mathbf{x}_i^T \mathbf{x}_j$$

Otherwise, put $S_{ij} = 0$. The weight matrix S of graph G models the local structure of the document space. We define D as a diagonal matrix whose entries are column (or row, since S is symmetric) sums of S, $D_{ii} = \sum_j S_{ji}$. We also define $L = D - S$, which is called the Laplacian matrix in spectral graph theory [17].

3. **Data Preprocessing and SVD Projection**: We remove the weighted mean of $\mathbf{x}$ from each $\mathbf{x}$

$$\hat{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}}, \quad \bar{\mathbf{x}} = \frac{1}{(\sum_i D_{ii})} \left(\sum_i \mathbf{x}_i D_{ii} \right)$$

and projected the document vector into the SVD subspace by throwing away those *zero* singular value.

$$\widehat{X} = U \Sigma V^T$$

where $\widehat{X} = [\hat{\mathbf{x}}_1, \cdots, \hat{\mathbf{x}}_n]$. We denote the transformation matrix of SVD by W_{SVD}, $W_{SVD} = U$. After SVD projection, the document vector $\hat{\mathbf{x}}$ becomes $\tilde{\mathbf{x}}$:

$$\tilde{\mathbf{x}} = W_{SVD}^T \hat{\mathbf{x}}$$

After this step, the term-document matrix X becomes $\widetilde{X} = [\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_n]$.

4. **LPP Projection**: Compute the eigenvectors and eigenvalues for the generalized eigen-problem:
$$\widetilde{X} L \widetilde{X}^T \mathbf{a} = \lambda \widetilde{X} D \widetilde{X}^T \mathbf{a} \tag{5.1}$$
Let $W_{LPP} = [\mathbf{a}_1, \cdots, \mathbf{a}_k]$ be the solutions of equation (5.1), ordered according to their eigenvalues, $\lambda_1 < \lambda_2 < \cdots < \lambda_k$. Thus, the embedding is as follows:
$$\mathbf{x} \to \mathbf{y} = W^T \hat{\mathbf{x}}$$
$$W = W_{SVD} W_{LPP} \quad \text{and} \quad \hat{\mathbf{x}} = \mathbf{x} - \frac{1}{(\sum_i D_{ii})} \left(\sum_i \mathbf{x}_i D_{ii} \right)$$
where $\mathbf{y}$ is a k-dimensional representation of the document $\mathbf{x}$. W is the transformation matrix.

5.3 Theoretical Analysis

5.3.1 Singular Value Decomposition

Let $X = [\mathbf{x}_1, \cdots, \mathbf{x}_m]^T$ be a $n \times m$ data matrix, $\mathbf{x}_i \in \mathbb{R}^n$. By SVD, X can be decomposed as follows:
$$X = USV^T \tag{5.2}$$
where $U = [\mathbf{u}_1, \cdots, \mathbf{u}_n]$ and $\mathbf{u}_i$ is the eigenvector of XX^T, $\mathbf{u}_i \in \mathbb{R}^n$. $V = [\mathbf{v}_1, \cdots, \mathbf{v}_m]$ and $\mathbf{v}_j$ is the eigenvector of $X^T X$, $\mathbf{v}_j \in \mathbb{R}^m$. S is a $n \times m$ matrix such that $\sigma_i = S_{ii}$ is the i^{th} largest singular value and $S_{ij} = 0$ for $i \neq j$. $\mathbf{u}_i$ and $\mathbf{v}_j$ are called left singular vectors and right singular vectors, respectively. The following points are well known or easily derived in matrix theory:

1. σ_i is the square root of the i^{th} largest eigenvectors of XX^T (or, $X^T X$).

2. U and V are both orthogonal matrices in that $U^T U = UU^T = I$ and $V^T V = VV^T = I$.

3. $\mathbf{u}_i$ forms a orthogonal basis for the input space $\mathbb{R}^n$. The matrix U can be thought of as a rotation transformation.

Clearly, the number of non-zero singular value is determined by the rank of X, say k. Thus, equation (5.2) can be rewritten as follows:

$$X = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_j^T \tag{5.3}$$

Thus, the document vectors can be reduced to a k-dimensional subspace spanned by $\{\mathbf{u}_1, \cdots, \mathbf{u}_k\}$ without losing any information. Also, it is easy to see that $\sum_{i=1}^{p} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$ is the best rank-p ($p \leq k$) approximation to X in terms of Frobenius matrix norm. Please see [6][21][57].

5.3.2 Geometry of the Document Space: Global VS. Local

From the last section, we see that the projection of LSI can be simply computed by using SVD. The left singular vectors $\mathbf{u}_i$ are the basis functions of the reduced space.

Let A be a transformation matrix. The low dimensional representation of X is $Y = A^X$. The optimal linear transformation which preserves inner product can be obtained by solving the following minimization problem:

$$\min_A \|X^T X - Y^T Y\|_F \tag{5.4}$$

where $\|M\|_F$ is the Frobenius matrix norm, $\|M\|_F = \sqrt{\sum_{ij} M_{ij}^2}$. By SVD, we have $X^T X = VSU^T USV^T = VS^2V^T$. Therefore, $Y = S_k V_k^T$ is the k-dimensional representation of X which best preserves inner product, where $S_k = diag(\sigma_1, \cdots, \sigma_k)$ and $V_k = [\mathbf{v}_1, \cdots, \mathbf{v}_k]$. Correspondingly, the optimal transformation preserving inner product is $A = U_k$ where $U_k = [\mathbf{u}_1, \cdots, \mathbf{u}_k]$.

LSI uses the matrix U_k to perform linear dimensionality reduction. In document analysis, inner product is one of the most frequently used similarity measures to discover the semantic structure of the document space. LSI preserves the global structure in spirit. However, in many real world applications, the local structure is more important especially when nearest neighbor search is involved. Moreover, LSI aims to discover the linear subspace on which the documents possibly reside. However, there is no convincing evidence that the document space is actually a linear subspace of the input space. A more naturally and reasonable assumption is that the document space is a sub-manifold embedded in the ambient space. It can be either linear or non-linear.

5.3.3 Connections to Latent Semantic Indexing

LPI is essentially obtained from a graph model. The graph structure represents the geometrical structure of the document space. In our algorithm, a nearest neighbor graph is constructed to discover the *local* manifold structure. Intuitively, LPI with a complete graph should discover the *global* structure. In this subsection, we present a theoretical analysis on the relationship between LPI and LSI. Specifically, we show that LPI with a complete graph is similar to LSI.

LSI is fundamentally based on SVD (Singular Value Decomposition). For a rank r term-document matrix $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$, LSI decompose the X using SVD as follow:

$$X = U\Sigma V^T$$

Where $\Sigma = diag(\sigma_1, \cdots, \sigma_r)$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$ are the singular values of X, $U = [\mathbf{u}_1, \cdots, \mathbf{u}_r]$ and $\mathbf{u}_i$ is called left singular vectors, $V = [\mathbf{v}_1, \cdots, \mathbf{v}_r]$ and $\mathbf{v}_i$ is called right singular vectors. LSI use the first k vectors in U as the transformation matrix to embed the original document into a k dimensional subspace. It can be easily checked that the column vectors of U are the eigenvectors of XX^T. Thus, LSI tries to solve the maximum eigenvalue problem:

$$XX^T\mathbf{a} = \lambda\mathbf{a}$$

In LPI, recall that the weight on an edge linking $\mathbf{x}_i$ and $\mathbf{x}_j$ is set to their inner product $\mathbf{x}_i^T\mathbf{x}_j$. Thus, the affinity matrix S of the complete graph can be written as X^TX. Since we first apply SVD to remove the components corresponding to the zero singular value, the matrix XX^T is of full rank. The generalized minimum eigenvalue problem of LPI can be written as follows:

$$\begin{aligned} & XLX^T\mathbf{a} = \lambda XDX^T\mathbf{a} \\ \Rightarrow\ & X(D-W)X^T\mathbf{a} = \lambda XDX^T\mathbf{a} \\ \Rightarrow\ & XWX^T\mathbf{a} = (1-\lambda)XDX^T\mathbf{a} \\ \Rightarrow\ & XX^TXX^T\mathbf{a} = (1-\lambda)XDX^T\mathbf{a} \end{aligned} \tag{5.5}$$

Since the diagonal matrix D is close to the identity matrix, $XDX^T \approx XX^T$. The *minimum* eigenvalues of equation (5.5) correspond to the *maximum* eigenvalues of the following equation:

$$XX^TXX^T\mathbf{a} = \lambda XX^T\mathbf{a}$$

Since XX^T is of full rank, we get:

$$XX^T\mathbf{a} = \lambda\mathbf{a}$$

which is just the eigenvalue problem of LSI. The above analysis shows that LPI with a complete graph is actually similar to LSI. Both of them discover the global structure. The only difference is that there is a diagonal matrix D in LPI which reflects the importance of the different document vectors. In LSI, every document vector is treated equally important. In LPI, the weight of document $\mathbf{x}_i$ is D_{ii}. We define $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i$ as the center vector of these document vectors. In complete graph situation, we have

$$\begin{aligned} D_{ii} &= \sum_{j=1}^n S_{ij} = \sum_{j=1}^n (X^TX)_{ij} = \sum_{j=1}^n \mathbf{x}_i^T\mathbf{x}_j \\ &= \mathbf{x}_i^T\sum_{j=1}^n \mathbf{x}_j = n\mathbf{x}_i^T\bar{\mathbf{x}} = \delta\mathbf{x}_i^T\frac{\bar{\mathbf{x}}}{||\bar{\mathbf{x}}||} \end{aligned}$$

where $\delta = n||\bar{\mathbf{x}}||$ is a constant. Note that all the $\mathbf{x}_i$ are normalized to 1, thus they are distributed on an unit hypersphere. $\bar{\mathbf{x}}/||\bar{\mathbf{x}}||$ is also on this unit hypersphere. Thus,

D_{ii} evaluates the cosine of the angle between vector $\mathbf{x}_i$ and $\bar{\mathbf{x}}$. In other words, the D_{ii} evaluates the cosine similarity between document $\mathbf{x}_i$ and the center. The closer to the center the document is, the larger weight it has. Some previous researches [81] show that such D will improve the result and our experiments will also show this.

5.3.4 Connections to Laplacian Eigenmap

In this subsection, we discuss the computational relationship between LPP and Laplacian Eigenmaps [10]. The eigenvalue problem of LPP scales with the number of dimensions (d), while that of Laplacian Eigenmaps scales with the number of data points (n). The rank of X is no greater than $min(n, d)$. Thus, if $d > n$, we can reduce the data space into an n dimensional subspace without losing any information by using Singular Value Decomposition (SVD). Correspondingly, the data matrix X in such a subspace becomes a square matrix. We have the following proposition:

Proposition 2. *If X is a full rank square matrix, then LPP and Laplacian Eigenmap have the same result.*

Proof. Recall that the eigenvalue problem of LPP is as follows:

$$XLX^T\mathbf{w} = \lambda XDX^T\mathbf{w} \tag{5.6}$$

Let $\mathbf{y} = X^T\mathbf{w}$. Equation (5.6) can be rewritten as follows:

$$XL\mathbf{y} = \lambda XD\mathbf{y} \tag{5.7}$$

Since X is a full rank square matrix, we get the following equation:

$$L\mathbf{y} = \lambda D\mathbf{y} \tag{5.8}$$

which is just the eigenvalue problem of Laplacian Eigenmaps. □ □

In many real world applications such as information retrieval, the dimensionality of the data space is typically much larger than the number of data points. In such

Table 5.1: 30 semantic categories from Reuters-21578 used in our experiments

category	**num of doc**	**category**	**num of doc**	**category**	**num of doc**
earn	3713	money-supply	87	iron-steel	37
acq	2055	gnp	63	ipi	36
crude	321	cpi	60	nat-gas	33
trade	298	cocoa	53	veg-oil	30
money-fx	245	alum	45	tin	27
interest	197	grain	45	cotton	24
ship	142	copper	44	bop	23
sugar	114	jobs	42	wpi	20
coffee	110	reserves	38	pet-chem	19
gold	90	rubber	38	livestock	18

Table 5.2: 30 semantic categories from TDT2 used in our experiments

category	**num of doc**	**category**	**num of doc**	**category**	**num of doc**
20001	1844	20048	160	20096	76
20015	1828	20033	145	20021	74
20002	1222	20039	141	20026	72
20013	811	20086	140	20008	71
20070	441	20032	131	20056	66
20044	407	20047	123	20037	65
20076	272	20019	123	20065	63
20071	238	20077	120	20005	58
20012	226	20018	104	20074	56
20023	167	20087	98	20009	52

a case, LPP and Laplacian Eigenmaps will have the same embedding result if these data vectors are linearly independent.

5.4 Experimental Results

5.4.1 Data Preparation

Reuters-21578

Reuters-21578 corpus[1] contains 21578 documents in 135 categories. In our experiments, we discarded those documents with multiple category labels, and selected the largest 30 categories. It left us with 8067 documents in 30 categories as described in table 5.1.

TDT2

The TDT2 corpus[2] consists of data collected during the first half of 1998 and taken from 6 sources, including 2 newswires (APW, NYT), 2 radio programs (VOA, PRI) and 2 television programs (CNN, ABC). It consists of 11201 on-topic documents which are classified into 96 semantic categories. In this dataset, we also removed those documents appearing in two or more categories and use the largest 30 categories thus leaving us with 9394 documents in 30 categories as described in table 5.2.

Each document is represented as a term-frequency vector. We simply removed the stop words and no further preprocessing was done.

5.4.2 Evaluation on Similarity Measure

The accuracy of similarity measure plays a crucial role in most of the information processing tasks, such as document clustering, classification, retrieval, etc. In this subsection, we evaluate the accuracy of similarity measure using two different indexing algorithms, i.e. LPI and LSI. The similarity measure we used is the cosine similarity.

From the <title> field of 300 TREC ad hoc topics (topic 251-550), we chose 30 keywords that appear in our data collection with highest frequencies, say, $q_i (i = 1, 2, \cdots, 30)$. For each keyword q_i, let D_i denote the set of the documents containing

[1]http://www.daviddlewis.com/resources/testcollections/reuters21578/

[2]http://www.nist.gov/speech/tests/tdt/tdt98/index.html

q_i. Let $D = D_1 \cup \cdots \cup D_{30}$. Finally, we get 30 document subsets and each subset contains multiple topics. Note that, these subsets are not necessarily disjoint. The numbers of documents of these 30 document subsets ranged from 226 to 802 with an average of 408, and the number of topics ranged from 12 to 20 with an average of 17.5. We removed the stop words. No further preprocessing was done. For the i^{th} subset, the documents are represented as vectors in a n_i dimensional vector space using the Term Frequency (TF) indexing scheme. The reason for generating such 30 document subsets is to split the data collection into small subsets so that we can compare our algorithm to LSI on each subset. In fact, the keywords can be thought of as queries in information retrieval. Thus, the comparison can be thought of as being performed on different queries.

For the original document set D, we compute its lower dimensional representations D_{LPI} and D_{LSI} by using LPI and LSI respectively. Similarly, D_{LPI} consists of 30 subsets, $D_{LPI,1}, \cdots, D_{LPI,30}$. D_{LSI} also consists of 30 subsets $D_{LSI,1}, \cdots, D_{LSI,30}$. We take the number of nearest neighbors for the LPI algorithm to be 7.

For each document subset D_i (or, $D_{LPI,i}$, $D_{LSI,i}$), we evaluate the similarity measure between the documents in D_i. Intuitively, we expect that similarity should be higher for any document pair related to the same topic (intra-topic pair) than for any pair related to different topics (cross-topic pair). Therefore, we adopted the average precision used in TREC [3], regarding an intra-topic pair as a relevant document and the similarity value as the ranking score. Specifically, we denote by p_i the document pair which has the i^{th} highest similarity value among all pairs of documents in the document set D_i. For each intra-topic pair p_k, its precision is evaluated as follows:

$$\text{precision}(p_k) = \frac{\#\text{ of intra-topic pairs } p_j \text{ where } j \le k}{k} \tag{5.9}$$

The average of the precision values over all intra-topic pairs in D_i was computed as the average precision of D_i. Note that, the definition of precision (Eqn. 5.9) we used here is the same as that used in [3].

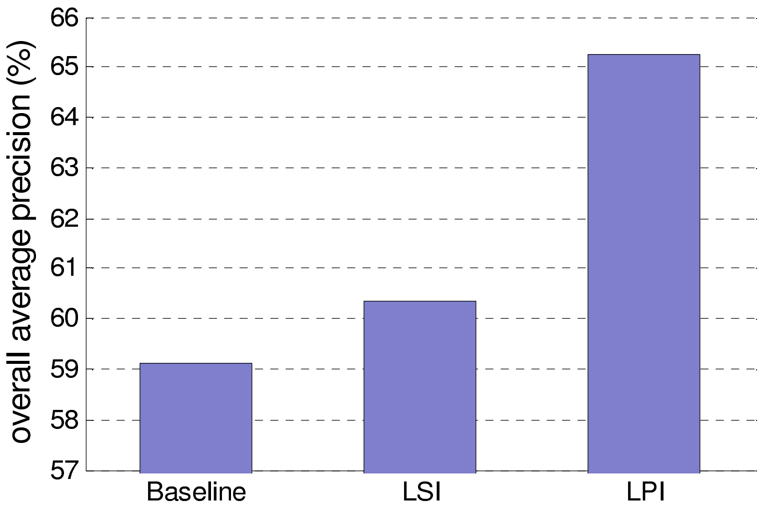


Figure 5.1: (Evaluation on similarity measure) The overall average precision using the Reuters-21578 document corpus.

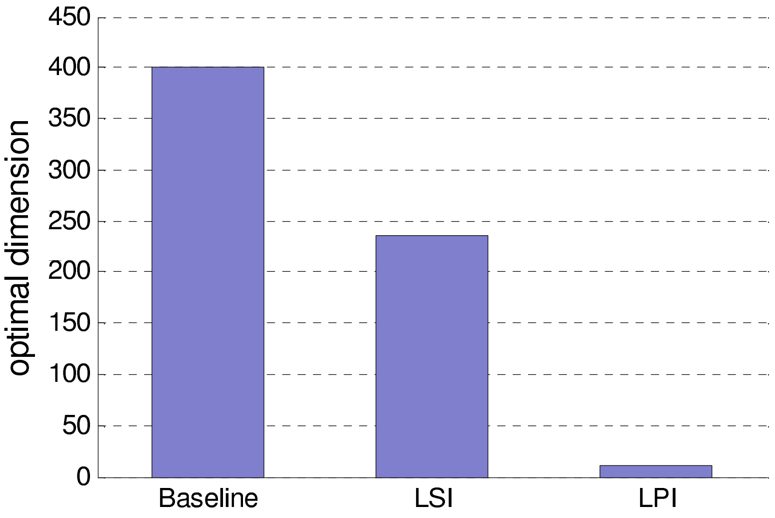


Figure 5.2: (Evaluation on similarity measure) The optimal dimensions of the baseline, LSI and LPI algorithms using the Reuters-21578 document corpus.

Table 5.3: Evaluation on similarity measure using the Reuters-21578 document corpus. The first column contains the keywords that are used to generate the document subsets. The LPI, LSI and baseline algorithms are compared. "AvP" stands for the average precision and "DR rate" stands for the dimensionality reduction rate.

	Baseline		LSI			LPI		
doc. subset	dims	AvP(%)	dims	DR rate	Δ AvP	dims	DR rate	Δ AvP
agreement	753	65.78	8	98.9	1.29	3	99.6	17.12
American	337	55.94	5	98.5	1.14	4	98.8	13.77
bank	780	36.91	6	99.2	6.05	9	98.8	12.81
control	226	78.95	209	7.5	0.59	11	95.1	-2.26
domestic	246	56.75	235	4.5	0.22	5	98	3.46
export	254	70.1	254	0	0	4	98.4	5.68
exports	295	53.74	282	4.4	0.08	7	97.6	5.43
five	761	72.42	538	29.3	0.71	59	92.2	-13.88
foreign	456	44.21	145	68.2	1	7	98.5	7.54
growth	319	46.16	72	77.4	3.09	4	98.7	9.7
income	333	81.83	177	46.8	0.89	59	82.3	-1.9
increase	548	46.77	506	7.7	0.1	16	97.1	-1.44
industrial	232	49.16	5	97.8	6.28	1	99.6	17.47
industry	361	50.67	357	1.1	0.02	4	98.9	13.47
international	679	51.08	6	99.1	1.36	4	99.4	20.91
investment	527	68.39	527	0	0	4	99.2	-1.38
losses	234	88.27	143	38.9	0.84	44	81.2	-0.68
money	247	53.11	231	6.5	0.06	9	96.4	-8.56
national	379	37	13	96.6	9.07	11	97.1	21.12
prices	551	57.16	546	0.9	0.01	5	99.1	8.75
production	335	57.85	325	3	0.04	7	97.9	12.01
public	282	56.3	282	0	0	3	98.9	-5.11
rates	300	51.65	286	4.7	0.19	3	99	-3.91
report	299	62.01	299	0	0	5	98.3	6.95
services	234	55.18	4	98.3	4.16	3	98.7	15.93
sources	256	52.74	252	1.6	0.03	6	97.7	12.18
talks	274	68.72	272	0.7	0	5	98.2	15.9
tax	594	95.67	251	57.7	0.16	34	94.3	-6.67
trade	550	47.09	512	6.9	0.3	4	99.3	-0.45
world	349	61.26	344	1.4	0.01	8	97.7	10.26

Results on Reuters 21578

We compared LPI (corresponding to document set D_{LPI}) to LSI (corresponding to document set D_{LSI}) and the original document representation (corresponding to document set D as baseline algorithm). In general, the performance of LPI and LSI varies with the number of dimensions. We showed the best results obtained by them. For each document subset, Table 5.4.2 lists the average precision and the dimensionality by using the baseline approach, and the improvements obtained by using LPI and LSI. In our experiments, the number of terms ($m > 6,000$) is larger than the number of documents (n). So, for the baseline approach, we reduced the document space to an n-dimensional subspace using SVD without losing any information. As can be seen, LPI achieved higher accuracy than LSI on 19 document subsets, while it failed on the other 11 subsets.

Figure 5.1 showed the average precision over the 30 document subsets. The overall average precisions for LPI, LSI and baseline algorithms are 64.78%, 60.22%, and 59.10%, respectively. LSI achieved little improvement (1.12%) over the baseline algorithm, while LPI achieved 5.68% improvement. Moreover, the average dimension in the original representation space of the 30 document subsets is 399.7. The average optimal dimensions for LSI and LPI are 246 and 12.7, respectively, as shown in Figure 5.2. By using LPI, we can obtain an extremely low dimensional representation for documents, which can facilitate some real world applications such as clustering, classification and retrieval.

Results on TDT2

In this test, we evaluate the accuracy of similarity measure in the LPI subspace, LSI subspace and the original representation space on the TDT2 database. Table 5.4.2 listed the average precision and dimensionality for each document subset. As can be seen, LPI achieved higher accuracy than LSI on all the document subsets.

Figure 5.3 shows the average precision over the 30 document subsets. The overall average precisions for LPI, LSI and baseline algorithms are 90.6%, 81.5%, and 81.4%, respectively. As can be seen, LPI significantly outperformed LSI on this database.

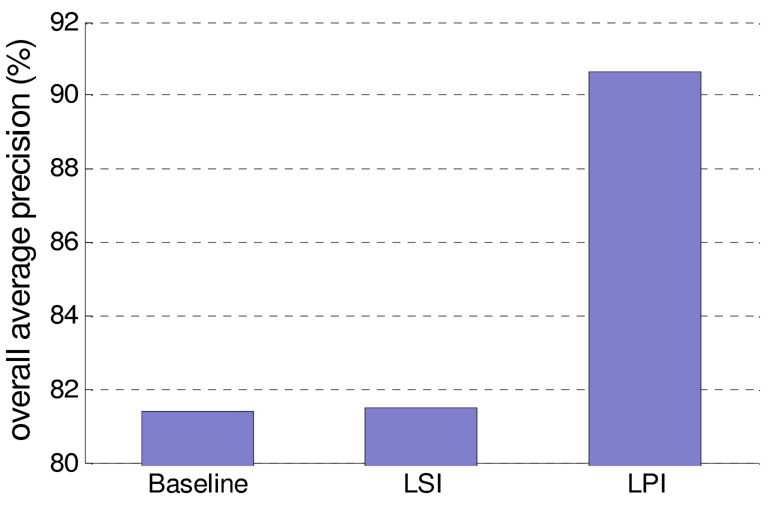


Figure 5.3: (Evaluation on similarity measure) The overall average precision using the TDT2 document corpus.

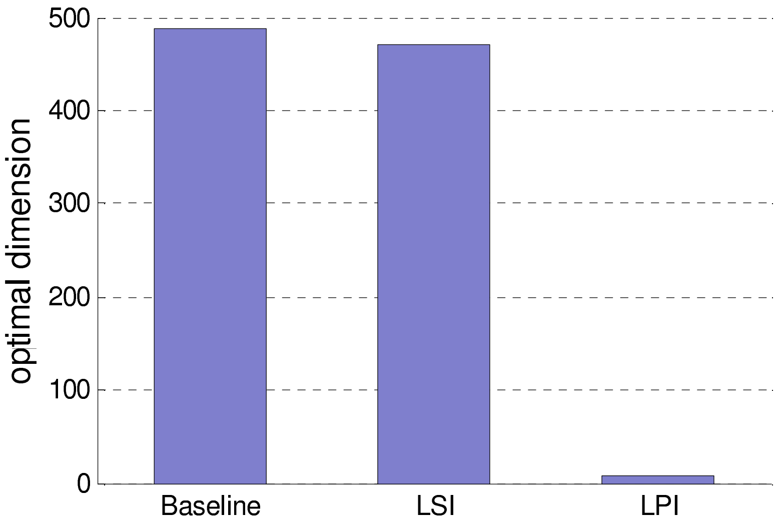


Figure 5.4: (Evaluation on similarity measure) The optimal dimensions of the baseline, LSI and LPI algorithms using the TDT2 document corpus.

Table 5.4: Evaluation on similarity measure using the TDT2 document corpus. The first column contains the keywords that are used to generate the document subsets. The LPI, LSI and baseline algorithms are compared. "AvP" stands for the average precision and "DR rate" stands for the dimensionality reduction rate.

	Baseline		LSI			LPI		
doc. subset	dims	AvP(%)	dims	DR rate	Δ AvP	dims	DR rate	Δ AvP
air	632	90.62	603	4.6	0.05	6	99.1	6.93
british	502	90.66	502	0	0	2	99.6	4.09
building	346	63.36	344	0.6	0.01	8	97.7	14.17
control	629	64.13	622	1.1	0	5	99.2	12.75
cooperation	337	91.73	337	0	0	4	98.8	5.55
court	715	93.91	713	0.3	0	7	99	4.39
decision	743	78.92	742	0.1	0	8	98.9	12.64
domestic	406	81.68	351	13.5	0.34	3	99.3	11.57
drug	272	91.87	254	6.6	0.37	3	98.9	6.87
fire	308	79.07	302	1.9	0.01	6	98.1	6.9
food	482	82.27	482	0	0	4	99.2	6.24
growth	554	92.34	359	35.2	1.01	49	91.2	1.23
health	429	83.82	424	1.2	0	8	98.1	8.68
history	419	73.62	386	7.9	0.47	8	98.1	21.11
human	383	65.4	372	2.9	0.04	6	98.4	13.42
impact	417	79.55	406	2.6	0.14	7	98.3	16.52
information	617	91.94	616	0.2	0	7	98.9	5.34
legal	542	94.61	541	0.2	0	6	98.9	3.99
material	735	80.99	735	0	0	10	98.6	10.47
money	770	70.43	770	0	0	6	99.2	18.25
peace	583	79.38	532	8.7	0.02	6	99	8.8
police	473	63.24	459	3	0.07	7	98.5	6.14
robert	337	85.38	337	0	0	5	98.5	10.73
russia	595	95.14	588	1.2	0	2	99.7	3.65
school	422	68.54	422	0	0	8	98.1	11.73
smoking	247	99	167	32.4	0.06	44	82.2	0.2
technology	332	78.9	332	0	0	3	99.1	8.15
trade	596	75.64	596	0	0	5	99.2	16.56
violence	337	72.66	337	0	0	7	97.9	11.66
women	499	83.08	478	4.2	0.09	7	98.6	9.52

The average optimal dimensions for LPI, LSI and baseline are 9.1, 469.6, and 488.6, respectively, as shown in Figure 5.4. LPI successfully encoded the discriminating information in a very low dimensional subspace.

5.4.3 Evaluation on Discriminating Power

As we described earlier, LPI is an unsupervised approximation to LDA algorithm which is supervised. In many cases, the data points (documents) may lack of labels. Specifically, for each data point, we do not know to what specific topic it is related to. However, it might be possible to discover the discriminant structure hidden in the data points. In other words, it might be possible to know if two data points are related to the same topic. It is often assumed that if two points x1, x2 are close in the intrinsic geometry of the data space, then they are related to the same topic. In this subsection, we evaluate the discriminating power of LPI and LSI. The data sets we used here is the same as that used in Section 5.4.2.

Let n denote the number of data points in the subset and c denote the number of semantic classes contained in this subset. For each semantic class, let n_i denote the number of data points in the i^{th} class. Let $\mathbf{x}_i^j$ denote the j^{th} sample in the i^{th} semantic class. For each data point $\mathbf{x}_i^j$, we find its n_i nearest neighbors in the subspace. Among these n_i data points, those sharing the same label as $textbfx_i^j$ are called relevant examples. Thus, we can compute the accuracy for $\mathbf{x}_i^j$ as follows:

$$\text{accuracy}(\mathbf{x}_i^j) = \frac{\text{relevant examples of } \mathbf{x}_i^j}{n_i} \tag{5.10}$$

Correspondingly, the average accuracy can be computed as follows:

$$\text{average accuracy} = \frac{1}{n}\sum_i \sum_j \text{accuracy}(\mathbf{x}_i^j) \tag{5.11}$$

Results on Reuters-21578

As before, the accuracy varies with the number of dimensions and we evaluate it at the optimal dimensionality. Table 5.4.3 shows the results for LPI, LSI and baseline

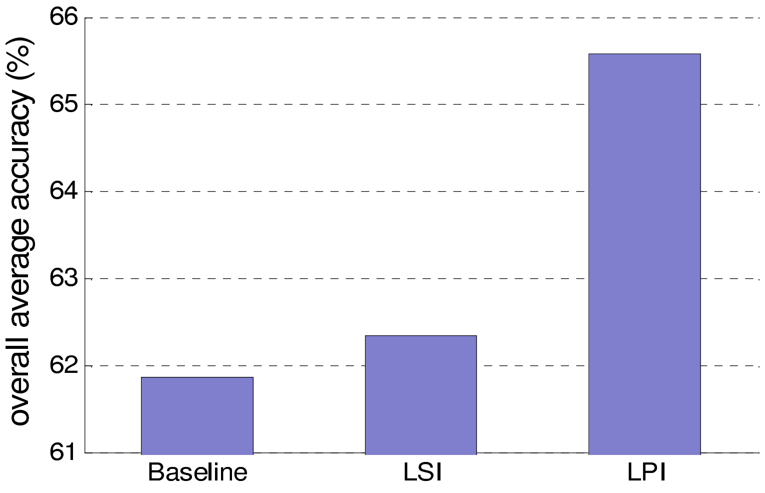


Figure 5.5: (Evaluation on discriminating power) The overall average accuracies of the baseline, LSI and LPI algorithms using the Reuters-21578 document corpus.

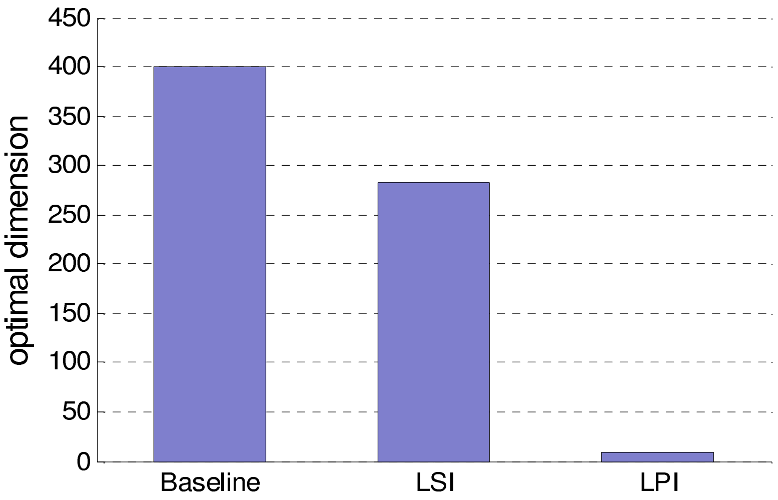


Figure 5.6: (Evaluation on discriminating power) The optimal dimensions of the baseline, LSI and LPI algorithms using the Reuters-21578 document corpus.

Table 5.5: Evaluation on discriminating power using the Reuters-21578 document corpus. The first column contains the keywords that are used to generate the document subsets. The LPI, LSI and baseline algorithms are compared. "AvP" stands for the average precision and "DR rate" stands for the dimensionality reduction rate.

	Baseline		LSI			LPI		
doc. subset	dims	AvP(%)	dims	DR rate	Δ AvP	dims	DR rate	Δ AvP
agreement	753	71.83	8	98.9	0.42	5	99.3	9.78
american	337	63.74	331	1.8	0.03	4	98.8	6
bank	780	46.25	8	99	1.89	9	98.8	3.07
control	226	77.43	212	6.2	0.27	4	98.2	1.3
domestic	246	58.92	7	97.2	1.9	5	98	4.49
export	254	67.3	254	0	0	4	98.4	3.61
exports	295	62.16	271	8.1	0.06	11	96.3	-1.3
five	761	60.9	691	9.2	0.01	19	97.5	-2.59
foreign	456	52.36	451	1.1	0	7	98.5	5.29
growth	319	52.41	12	96.2	0.47	8	97.5	5.87
income	333	76.33	301	9.6	0.04	24	92.8	4.13
increase	548	51.01	548	0	0	11	98	0.19
industrial	232	63.08	10	95.7	1.83	4	98.3	5.83
industry	361	58.93	354	1.9	0.03	5	98.6	4.34
international	679	61.1	669	1.5	0.01	6	99.1	11.77
investment	527	66.05	527	0	0	2	99.6	-0.7
losses	234	77.38	139	40.6	0.34	29	87.6	1.57
money	247	51.2	221	10.5	0.04	9	96.4	-3.37
national	379	55.97	12	96.8	1.58	11	97.1	8.76
prices	551	60	542	1.6	0.01	4	99.3	6.7
production	335	58.2	327	2.4	0.01	3	99.1	5.16
public	282	56.47	275	2.5	0.02	4	98.6	-1.98
rates	300	54.42	285	5	0.15	4	98.7	-4.14
report	299	58.33	294	1.7	0.12	5	98.3	5.2
services	234	59.89	4	98.3	1.74	3	98.7	9.15
sources	256	58.83	7	97.3	3.51	6	97.7	8.74
talks	274	75.58	272	0.7	0.01	5	98.2	7.69
Tax	594	80.94	594	0	0	9	98.5	1.3
trade	550	54.93	521	5.3	0.1	15	97.3	1.22
world	349	63.63	343	1.7	0.01	5	98.6	4.73

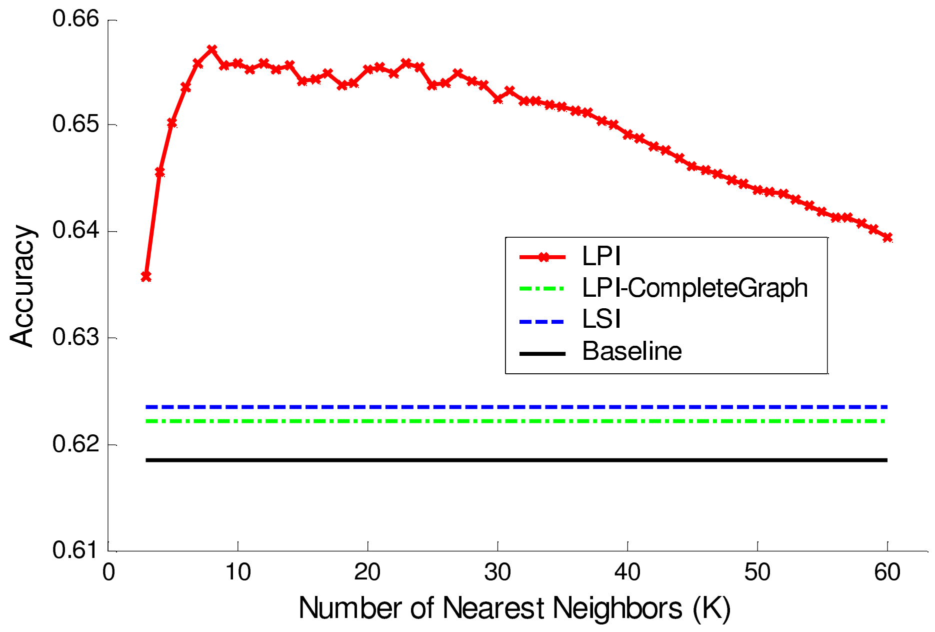


Figure 5.7: Discriminating power of LPI VS. the number of nearest neighbors using the Reuters-21578 document corpus.

algorithm on each document subsets, and Figure 5.5 shows the average accuracies. As can be seen, LPI outperforms LSI on 24 subsets. On the other 6 subsets, LSI works better. This shows that LPI is much powerful than LSI as to discovering the discriminant structure hidden in the document space. This is mainly due to fact that LPI provides an optimal approximation to the Linear Discriminant Analysis algorithm which aims to discover the discriminant structure in a supervised manner. Figure 5.6 shows the average optimal dimensionality obtained by baseline, LSI and LPI. The dimensionality obtained by LPI is much lower than that obtained by LSI.

In the LPI algorithm, one needs to set the number of nearest neighbors (k), which defines the "locality". As we show in Section 5.3.3, LPI approximates LSI when k tends to be infinity. In Figure 5.7, we show the relationship between the average accuracy and the value of k. Here, the average accuracy is the average of the accuracies for the 30 subsets. The value of k varies from 3 to the number of documents. As can be seen, the performance of LPI reaches its peak when k is 8. As k increases, the performance decreases. The performance of LPI with a complete

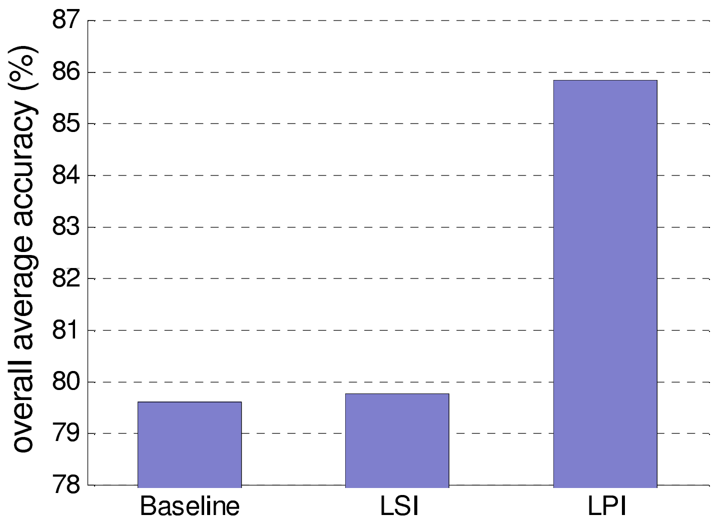


Figure 5.8: (Evaluation on discriminating power) The overall average accuracies of the baseline, LSI and LPI algorithms using the TDT2 document corpus.

graph (k is infinity) is close to that of LSI. This test shows that the local structure is more important than the global structure as to discovering the discriminant structure of the document space.

Results on TDT2

Table 5.4.3 shows the performance comparisons on the 30 document subsets from the TDT2 database, and Figure 5.8 shows the average accuracies of the baseline (79.6), LSI (79.8) and LPI (85.8) algorithms. As can be seen, LPI works better than LSI on all the 30 subsets. The performance of LSI is actually very close to the baseline, while LPI can get more improvement. Also, the optimal dimensionality obtained by LPI (7.1) is much lower than that obtained by LSI (436.0), as shown in Figure 5.9.

Figure 5.10 shows the performance of LPI as a function of the number of nearest neighbors. The best performance is obtained when k is 6. The performance of LSI is close to baseline, and LPI with a complete graph (k is infinity) works a little bit better than LSI. Note that, when k is infinity, LPI no longer discovers the local structure but the global structure, as we show in Section 5.3.3.

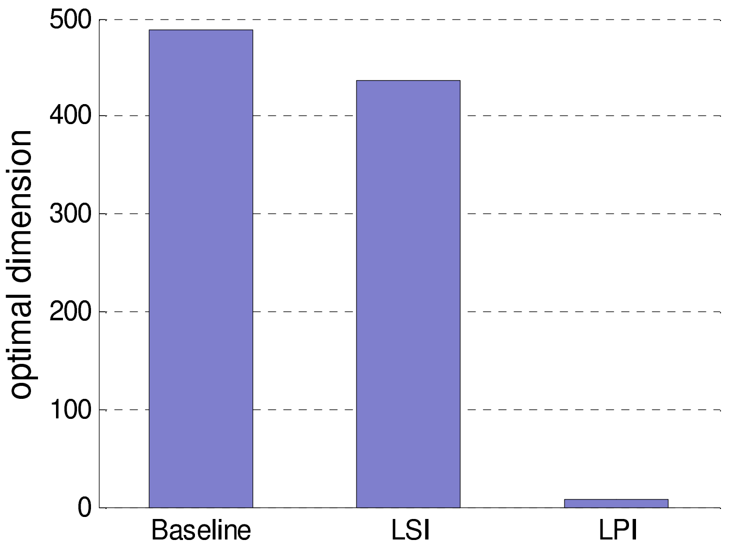


Figure 5.9: (Evaluation on discriminating power) The optimal dimensions of the baseline, LSI and LPI algorithms using the TDT2 document corpus.

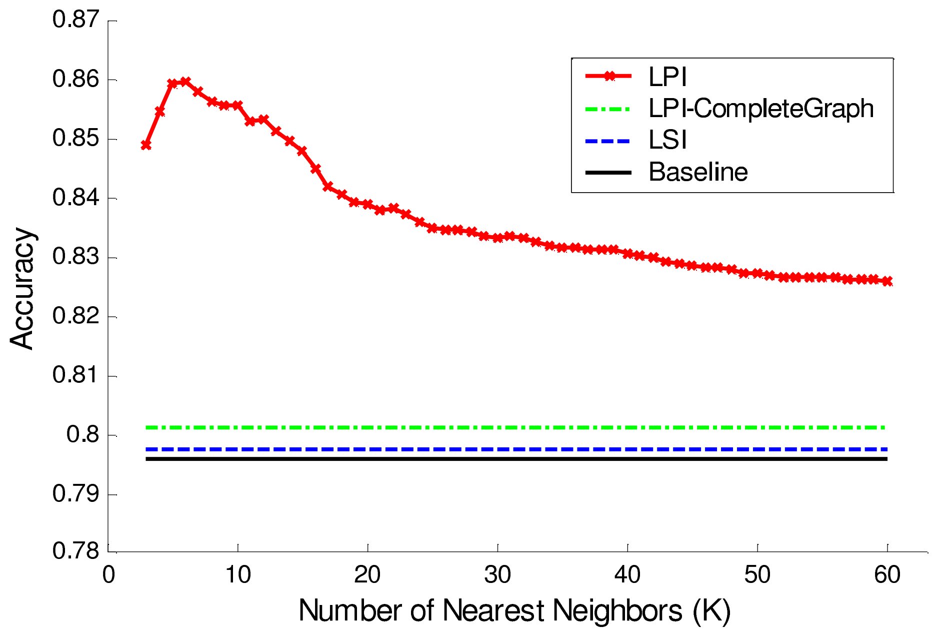


Figure 5.10: Discriminating power of LPI VS. the number of nearest neighbors using the TDT2 document corpus.

Table 5.6: Evaluation on discriminating power using the TDT2 document corpus. The first column contains the keywords that are used to generate the document subsets. The LPI, LSI and baseline algorithms are compared. "AvP" stands for the average precision and "DR rate" stands for the dimensionality reduction rate.

	Baseline		LSI			LPI		
doc. subset	dims	AvP(%)	dims	DR rate	Δ AvP	dims	DR rate	Δ AvP
air	632	82.51	611	3.3	0.01	6	99.1	6.83
british	502	81.55	489	2.6	0	3	99.4	6.74
building	346	71.79	10	97.1	0.28	9	97.4	6.03
control	629	72.1	608	3.3	0.02	9	98.6	5.92
cooperation	337	84.8	337	0	0	6	98.2	6.64
court	715	86.39	701	2	0.01	9	98.7	4.05
decision	743	77.78	733	1.3	0	7	99.1	8.56
domestic	406	80.54	403	0.7	0	3	99.3	5.94
drug	272	86.33	262	3.7	0.01	3	98.9	4.71
fire	308	76.67	301	2.3	0.01	6	98.1	3.99
food	482	78.23	475	1.5	0	5	99	5.1
growth	554	87.33	554	0	0	24	95.7	2.33
health	429	78.58	423	1.4	0	7	98.4	6.75
history	419	77.8	399	4.8	0.01	9	97.9	9.41
human	383	66.94	10	97.4	0.76	6	98.4	7.82
impact	417	80	406	2.6	0.05	7	98.3	7.4
information	617	83.85	605	1.9	0	7	98.9	8.16
legal	542	86.35	539	0.6	0	5	99.1	4.34
material	735	81.36	718	2.3	0	11	98.5	8.43
money	770	74.24	770	0	0	7	99.1	10.16
peace	583	74.19	561	3.8	0.01	6	99	6.34
police	473	69.09	458	3.2	0.07	7	98.5	8.57
robert	337	85.61	320	5	0.05	5	98.5	6.27
russia	595	88.72	4	99.3	4.02	2	99.7	7.68
school	422	71.33	415	1.7	0.02	7	98.3	1.64
smoking	247	95.96	227	8.1	0.03	19	92.3	0.59
technology	332	78.48	332	0	0	3	99.1	5.32
trade	596	79.38	591	0.8	0	5	99.2	9.01
violence	337	70.41	329	2.4	0.02	3	99.1	7.72
women	499	79.16	489	2	0.01	8	98.4	4.44

5.4.4 2D Visualization of the Document Set

Our theoretical analysis shows that LPI is able to map the documents related to the same topic as close to each other as possible. This motivates us to perform traditional clustering in the LPI subspace rather than directly in the original space. In this subsection, we first present some embedding results by using LPI and LSI.

Figure (5.11) and (5.12) shows the 2-D embedding results on the TDT2 corpus. The experiments were conducted on 5, 6, 7, 8, 9 and 10 classes, respectively. As can be seen, LPI was more powerful than LSI as to separating the documents with different semantics.

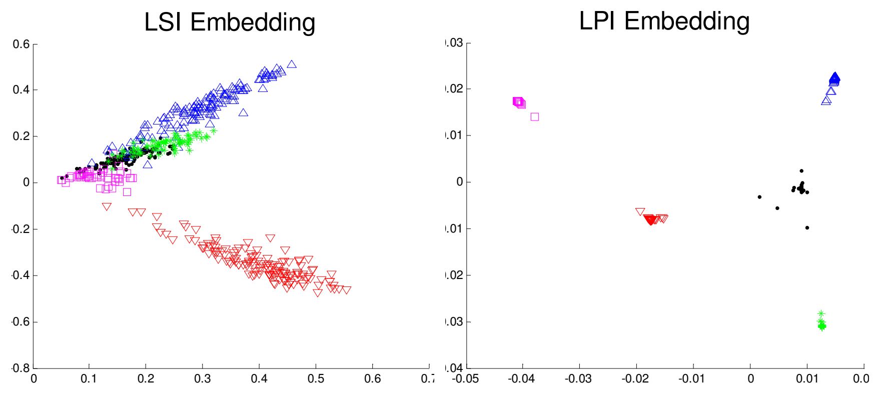
5.4.5 Document Clustering

In this subsection, we investigate the use of LPI for document clustering. Given a set of documents in k classes, we first project them into a $(k-1)$-dimensional subspace and then apply traditional clustering method in the subspace. In our experiments, we simply use the k-means clustering algorithm.

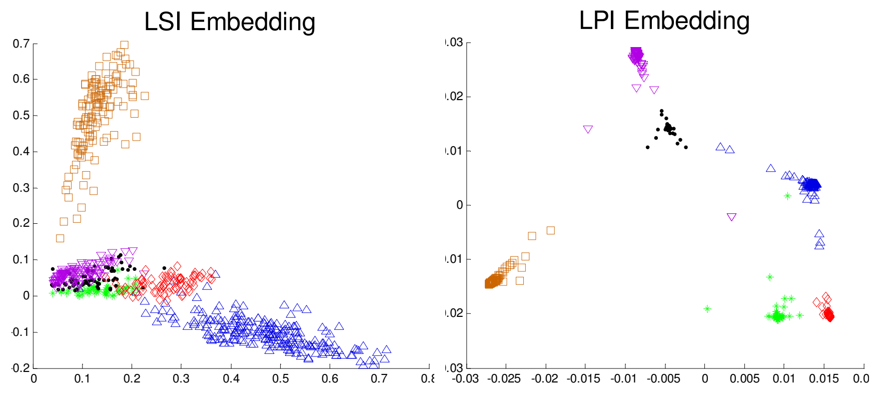
To demonstrate how our method improves the performance of document clustering, we compared five methods on two data sets, i.e. Reuters-21578 and TDT2. These five methods are listed below:

- K-means on original term-document matrix (K-means) - This method is treated as our baseline.
- K-means after LSI (LSI)
- K-means after LPI (LPI)
- Spectral Clustering (K-means after Laplacian Eigenmaps, or LE)
- Non-negative Matrix Factorization based clustering (NMF-NCW, [81])

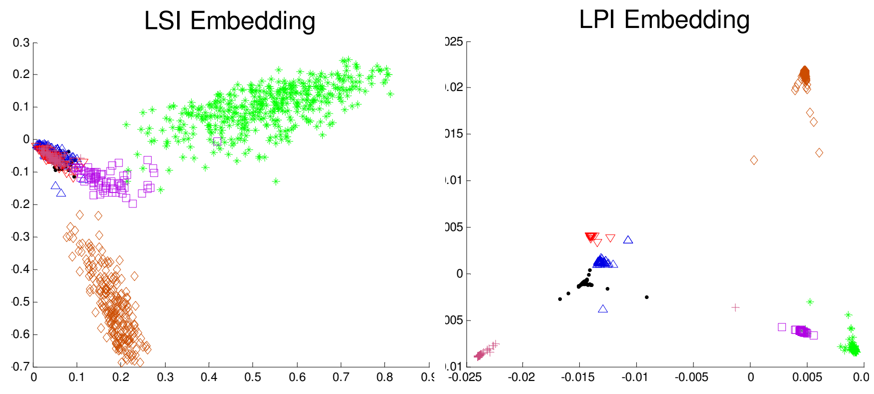
Note that, the two methods LPI and LE need to construct a graph on the documents. In this experiment, we used the same graph for these two methods and the number of nearest neighbors was set to 15. The weighted Non-negative Matrix Factorization



(a) 5 classes

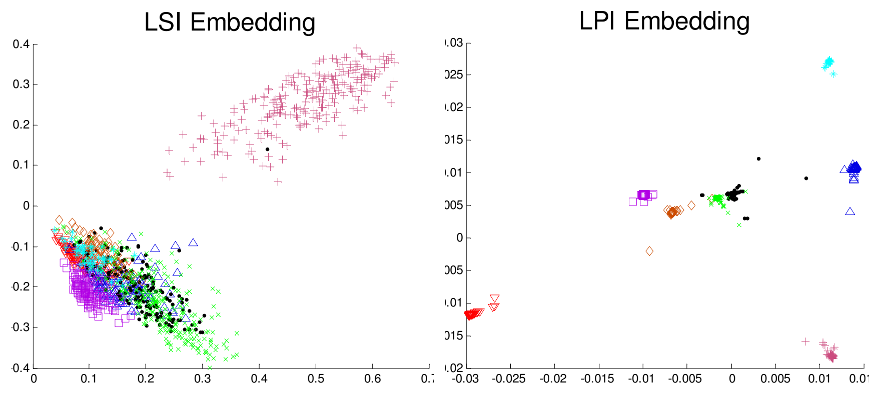


(b) 6 classes

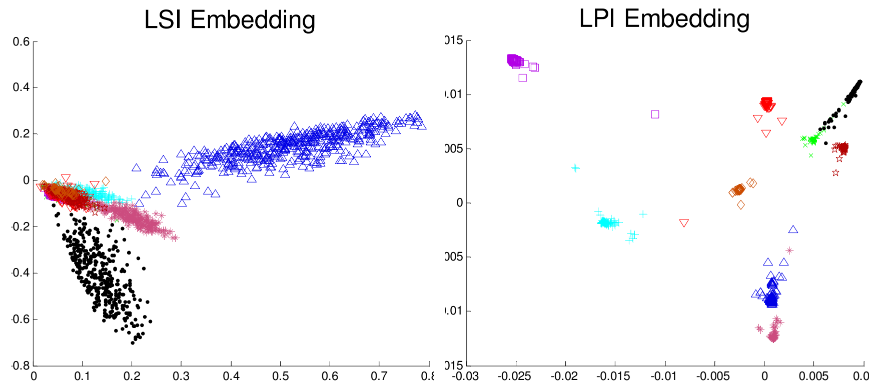


(c) 7 classes

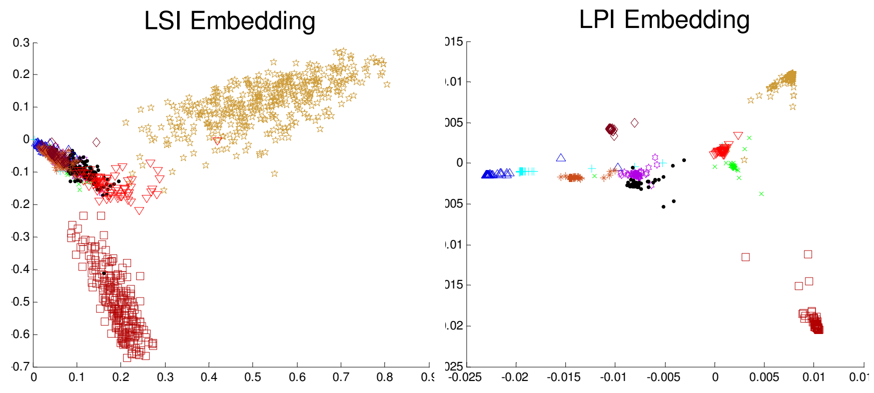
Figure 5.11: 2D Embedding results of LSI and LPI on the TDT2 corpus. Each color represents a topic.



(a) 8 classes



(b) 9 classes



(c) 10 classes

Figure 5.12: 2D Embedding results of LSI and LPI on the TDT2 corpus. Each color represents a topic.

Table 5.7: Performance comparisons on TDT2 corpus

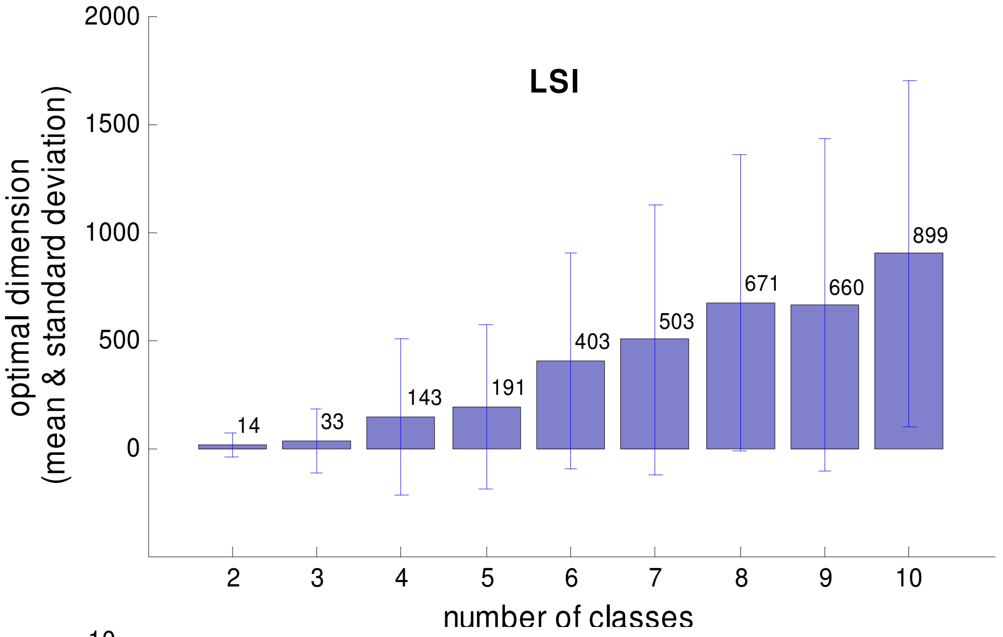
	Accuracy						
k	Kmeans	LSI(best)	LSI	LPI(best)	LPI	LE	NMF-NCW
2	0.989	0.992	0.977	0.998	0.998	0.998	0.985
3	0.974	0.985	0.944	0.996	0.996	0.996	0.953
4	0.959	0.970	0.894	0.996	0.996	0.996	0.964
5	0.948	0.961	0.914	0.993	0.993	0.993	0.980
6	0.945	0.954	0.879	0.993	0.992	0.992	0.932
7	0.883	0.903	0.849	0.990	0.988	0.987	0.921
8	0.874	0.890	0.829	0.989	0.987	0.988	0.908
9	0.852	0.870	0.810	0.987	0.983	0.984	0.895
10	0.835	0.850	0.786	0.982	0.979	0.978	0.898
ave.	0.918	0.931	0.876	0.992	0.990	0.990	0.937

	Mutual Information						
k	Kmeans	LSI(best)	LSI	LPI(best)	LPI	LE	NMF-NCW
2	0.962	0.965	0.925	0.981	0.981	0.981	0.939
3	0.946	0.962	0.894	0.977	0.976	0.976	0.924
4	0.932	0.942	0.856	0.979	0.979	0.979	0.951
5	0.935	0.942	0.892	0.975	0.973	0.973	0.965
6	0.936	0.939	0.878	0.975	0.974	0.974	0.923
7	0.884	0.892	0.849	0.969	0.968	0.966	0.915
8	0.889	0.895	0.841	0.970	0.967	0.967	0.911
9	0.875	0.878	0.831	0.970	0.966	0.967	0.905
10	0.865	0.869	0.813	0.962	0.959	0.958	0.897
ave.	0.914	0.920	0.864	0.973	0.971	0.97	0.926

based document clustering algorithm (NMF-NCW, [81]) is a recently proposed algorithm, which has shown to be very effective in document clustering. Please see [81] for details.

Table 5.7 and 5.8 shows the experimental results on the TDT2 and the Reuters corpus, respectively. The evaluations were conducted with different number of clusters, ranging from 2 to 10. For each given cluster number k, 50 tests were conducted on different randomly chosen clusters, and the average performance was computed over these 50 tests. For each test, K-means algorithm was applied 10 times with different start points and the best result in terms of the objective function of K-means was recorded.

After LSI, LPI or Laplacian Eigenmaps, how to determine the dimensions of the



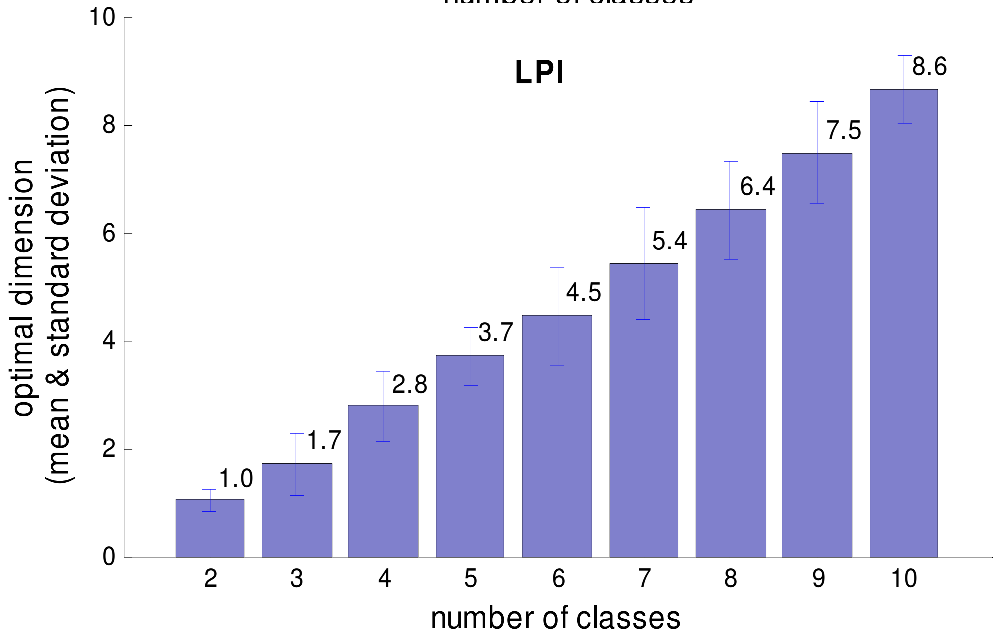
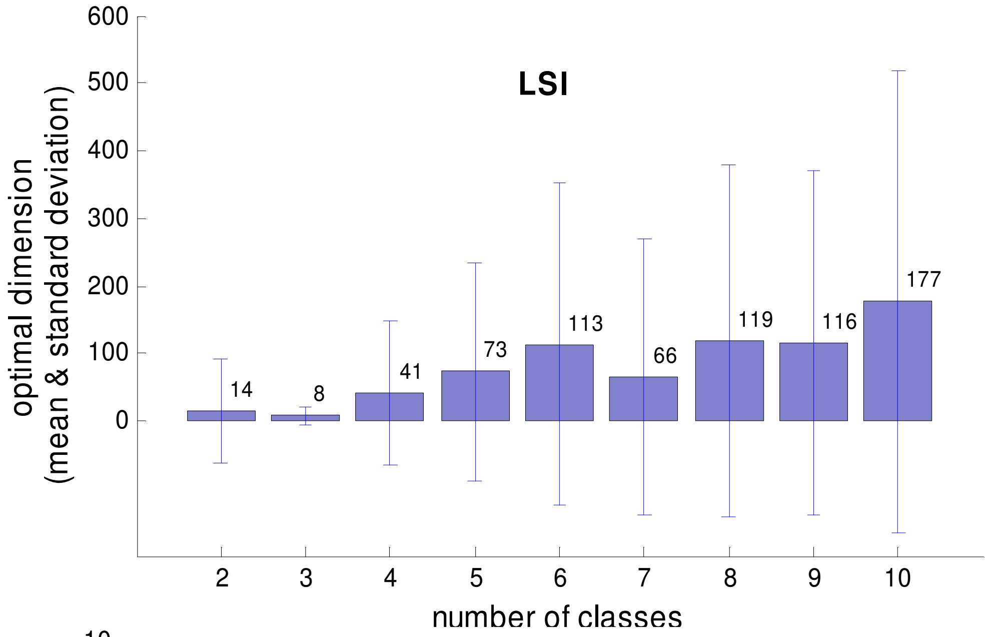


Figure 5.13: Optimal dimension with different number of classes on TDT2 corpus. Each bar shows the average of 50 test runs, and the error bar indicates the standard deviation.



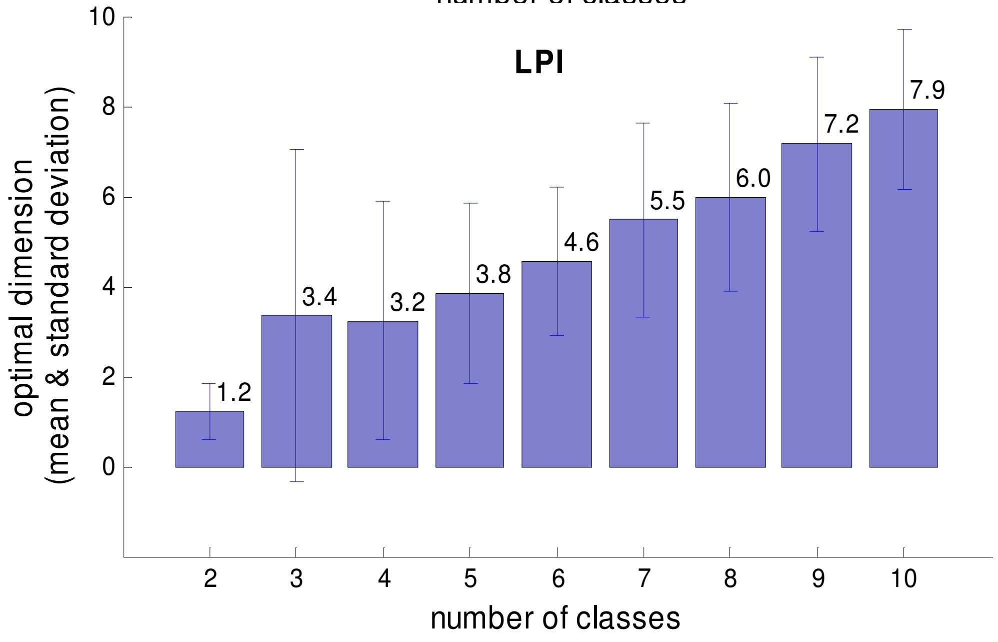


Figure 5.14: Optimal dimension with different number of classes on Reuters Corpus. Each bar shows the average of 50 test runs, and the error bar indicates the standard deviation.

Table 5.8: Performance comparisons on Reuters corpus

	Accuracy						
k	Kmeans	LSI(best)	LSI	LPI(best)	LPI	LE	NMF-NCW
2	0.871	0.913	0.864	0.963	0.923	0.923	0.925
3	0.775	0.815	0.768	0.884	0.816	0.816	0.807
4	0.732	0.773	0.715	0.843	0.793	0.793	0.787
5	0.671	0.704	0.654	0.780	0.737	0.737	0.735
6	0.655	0.683	0.642	0.760	0.719	0.719	0.722
7	0.623	0.651	0.610	0.724	0.694	0.694	0.689
8	0.582	0.617	0.572	0.693	0.650	0.650	0.662
9	0.553	0.587	0.549	0.661	0.625	0.625	0.623
10	0.545	0.573	0.540	0.646	0.615	0.615	0.616
ave.	0.667	0.702	0.657	0.773	0.730	0.730	0.730

	Mutual Information						
k	Kmeans	LSI(best)	LSI	LPI(best)	LPI	LE	NMF-NCW
2	0.600	0.666	0.569	0.793	0.697	0.697	0.705
3	0.567	0.594	0.536	0.660	0.601	0.601	0.600
4	0.598	0.621	0.573	0.671	0.635	0.635	0.634
5	0.563	0.567	0.538	0.633	0.603	0.603	0.587
6	0.579	0.587	0.552	0.636	0.615	0.615	0.603
7	0.573	0.572	0.547	0.629	0.617	0.617	0.600
8	0.556	0.557	0.530	0.615	0.587	0.587	0.583
9	0.549	0.545	0.532	0.605	0.586	0.586	0.560
10	0.552	0.549	0.528	0.607	0.586	0.586	0.561
ave.	0.571	0.584	0.545	0.650	0.614	0.614	0.604

subspace is still an open problem. In k cluster situation, we choose the first k-1 dimensions in LPI based on our previous analysis. For Laplacian Eigenmaps, since the first eigenvector is $\mathbf{1}$, we use the following k-1 dimensions. Note that, in typical spectral clustering, the dimension of the subspace is set to the number of clusters [56] which is the same as our selection in spirit (note that, [56] does not remove the first eigenvector, i.e. $\mathbf{1}$). For LSI, we choose the k dimensions for comparison. Besides such determined dimension, for LSI and LPI, we also compute their best performances on different dimensions in each test. We iterate all the dimensions for the best clustering performance and average all the 50 best results.

In table 5.7 and 5.8, LSI, LPI and LE indicate the results obtained with predetermined dimension, while "LSI (best)" and "LPI (best)" are the best result. Figure

(5.13) and (5.14) shows the optimal dimensions with different number of clusters by using LPI and LSI. The optimal dimension in LSI is much higher than LPI. Also, the variance of the dimensions obtained by using LSI is much higher than that obtained by using LPI. For LPI, the optimal number is nearly k-1, where k is the number of clusters. This figure shows that LPI is more powerful than LSI as to finding the intrinsic dimensionality of the document space. Therefore, LPI is very suitable for clustering.

The experimental results show that LSI seems not to be promising for document clustering, because the k-means in the LSI subspace is even worse than that in the original document space. As can be seen, LPI performs much better than LSI. We also see that, the result of Laplacian Eigenmaps is almost identical to the result of LPI. Actually, in our experiments, for 312 out of 450 (50×9) tests on Reuters corpus and 430 out of 450 (50×9) tests on TDT2 corpus, the matrix X is a square matrix of full rank. Thus the embedding results using LPI are identical to those using Laplacian Eigenmaps.

In [81], Xu et. al compared the NFM-NCW method with the spectral clustering method. In their comparison, they construct the affinity matrix in spectral clustering as a complete graph. While in our LSI and LE methods, the p-nearest neighbor graphes which put more focus on the local geometric document structure were used. More experiments on the different graph construction will be given in the next subsection.

CHAPTER 6
FEATURE SELECTION USING LAPLACIAN SCORE

Many problems in information processing involve some form of dimensionality reduction. Feature selection that chooses the important original features is an effective dimensionality reduction technique. An important feature for a learning task can be defined as one whose removal degrades the learning accuracy. The irrelevant features must be removed to allow the learning algorithm to focus on only the relevant features. Most of works concerning feature selection have been carried out under the supervised learning paradigm [28][75][78], paying little attention to unsupervised learning [24] [60]. Supervised feature selection algorithms are used when class labels of the data are available, otherwise unsupervised feature selection algorithm are employed.

Feature selection methods can be classified into "wrapper" methods and "filter" methods [42]. The wrapper model techniques evaluate the features using the learning algorithm that will ultimately be employed. Thus, they "wrap" the selection process around the learning algorithm. Most of the feature selection methods are wrapper methods. Algorithms based on the filter model examine intrinsic properties of the data to evaluate the features prior to the learning tasks. The filter based approaches almost always rely on the class labels, most commonly assessing correlations between features and the class label. In this paper, we are particularly interested in the filter methods. Some typical filter methods include data variance, Pearson correlation coefficients, Fisher criterion score, and Kolmogorov-Smirnov test.

Most of the existing filter methods are supervised. Data variance might be the most simple yet effective *unsupervised* evaluation of the features. The variance along a dimension reflects its representative power. Data variance can be used as a criteria for feature selection and extraction. For example, Principal Component Analysis (PCA) is a classical feature extraction method which finds a set of mutually orthogonal basis functions that capture the directions of maximum variance in the data.

Although the data variance criteria finds features that are useful for representing data, there is no reason to assume that these features must be useful for discriminating between data in different classes. Fisher criterion score seeks features that are efficient for discrimination [22]. It assigns the highest score to the feature on which the data points of different classes are far from each other while requiring data points of the same class to be close to each other. Fisher criterion can be also used for feature extraction, such as Linear Discriminant Analysis (LDA).

In this paper, we introduce a novel feature selection algorithm called **Laplacian Score** (LS). For each feature, its Laplacian Score is computed to reflect its locality preserving power. LS is based on the observation that, two data points are probably related to the same topic if they are close to each other. In fact, in many learning problems such as classification, the local structure of the data space is more important than the global structure. In order to model the local geometric structure, we construct a nearest neighbor graph. LS seeks those features that respect this graph structure.

It is worthwhile to highlight several aspects of the proposed approach here:

1. Our algorithm for feature selection can be performed in either supervised or unsupervised manner. This is reflected by the way that we construct the nearest neighbor graph. When the label information is available, it can be incorporated into the graph.

2. Laplacian score is independent to the learning tasks. This makes it a general data preprocessing method.

3. Laplacian score is easy to implement and computationally very efficient.

6.1 Feature Selection VS. Feature Extraction

Feature selection and extraction cope with the problem of excessive dimensionality of the data space. Feature selection reduces the dimensionality by selecting a feature subsets, while feature extraction reduces the dimensionality by combining features. However, they may share the same objective function.

The simplest criterion might be maximizing the data variance. Suppose we have m data points, $\mathbf{x}_1, \cdots, \mathbf{x}_m$. The objective function can be written as follows:

$$\max_{f} \sum_{i=1}^{m} \left(f(\mathbf{x}_i) - \mu\right)^2 \tag{6.1}$$

$$\mu = \frac{1}{m} \sum_{i=1}^{m} f(\mathbf{x}_i) \tag{6.2}$$

Different from data variance which can be computed without label information, Fisher criterion score is supervised. Fisher criterion score evaluate features according to their discriminating power. In the two class case for each feature r one computes the value

$$F_r = \frac{(\mu_1 - \mu_2)^2}{n_1\sigma_1^2 + n_2\sigma_2^2} \tag{6.3}$$

where μ_1 is the mean value of class 1 corresponding to the r-th feature and μ_2 the mean value in class 2, and σ_1 and σ_2 are the standard deviations of class 1 and class 2. n_1 and n_2 are the numbers of data points in the two classes, respectively. The idea behind this is, that the more a feature can separate the two classes, the better it is. For multi-class case, one can easily generalize this to

$$F_r = \frac{\sum_{i=1}^{c} n_i(\mu_i - \mu)^2}{\sum_{i=1}^{c} n_i\sigma_i^2} \tag{6.4}$$

where c is the number of classes, n_i is the number of data points in the i-th class, μ_i is the mean of the i-th class, and σ_i is the variance of the i-th class.

Both data variance and Fisher criterion score are filter methods and independent to the learning tasks. Data variance is unsupervised and aims to discover the geometrical structure of the data space, while Fisher criterion is supervised and aims to discover the discriminative structure. In the following section, we introduce a new feature selection algorithm called **Laplacian Score**, which can be performed in either supervised or unsupervised manner and can discover both geometrical and discriminative structure.

6.2 Laplacian Score

Laplacian Score (LS) is fundamentally based on Laplacian Eigenmaps [10] and Locality Preserving Projection [33]. The basic idea of LS is to evaluate the features according to their locality preserving power.

6.2.1 *The Algorithm*

Let L_r denote the Laplacian Score of the r-th feature. Let f_{ri} denote the i-th sample of the r-th feature, $i = 1, \cdots, m$. Our algorithm can be stated as follows:

1. Construct a nearest neighbor graph G with m nodes. The i-th node corresponds to $\mathbf{x}_i$. We put an edge between nodes i and j if $\mathbf{x}_i$ and $\mathbf{x}_j$ are "close", i.e. $\mathbf{x}_i$ is among k nearest neighbors of $\mathbf{x}_j$ or $\mathbf{x}_j$ is among k nearest neighbors of $\mathbf{x}_i$. When the label information is available, one can put an edge between two nodes sharing the same label.

2. If nodes i and j are connected, put

$$S_{ij} = \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \cdot \|\mathbf{x}_j\|} \tag{6.5}$$

   Otherwise, put $S_{ij} = 0$. The weight matrix S of the graph models the local structure of the data space.

3. For the r-th feature, we define:

$$\mathbf{f}_r = [f_{r1}, f_{r2}, \cdots, f_{rm}]^T$$

$$D = diag(S\mathbf{1}), \mathbf{1} = [1, \cdots, 1]^T$$

$$L = D - S$$

   where the matrix L is often called graph Laplacian [17]. Let

$$\tilde{\mathbf{f}}_r = \mathbf{f}_r - \frac{\mathbf{f}_r^T D\mathbf{1}}{\mathbf{1}^T D\mathbf{1}}\mathbf{1}$$

4. Compute the Laplacian Score of the r-th feature as follows:

$$L_r = \frac{\tilde{\mathbf{f}}_r^T L \tilde{\mathbf{f}}_r}{\tilde{\mathbf{f}}_r^T D \tilde{\mathbf{f}}_r} \tag{6.6}$$

6.3 Theoretical Analysis

In this section, we provide theoretical justifications for our algorithm.

6.3.1 Objective Function

Recall that given a data set we construct a weighted graph G with edges connecting nearby points to each other. S_{ij} evaluates the similarity between the *i-th* and *j-th* nodes. Thus, the importance of a feature can be thought of as the degree it respects the graph structure. To be specific, a "good" feature should the one on which two data points are close to each other if and only if there is an edge between these two points. A reasonable criterion for choosing a good feature is to minimize the following object function:

$$L_r = \frac{\sum_{ij}(f_{ri} - f_{rj})^2 S_{ij}}{Var(\mathbf{f}_r)} \tag{6.7}$$

where $Var(\mathbf{f}_r)$ is the estimated variance of the r-th feature. By minimizing $\sum_{ij}(f_{ri} - f_{rj})^2 S_{ij}$, we prefer those features respecting the pre-defined graph structure. For a good feature, the bigger S_{ij}, the smaller $(f_{ri} - f_{rj})$, and thus the Laplacian Score tends to be small. Following some simple algebraic steps, we see that

$$\begin{aligned}
& \sum_{ij} (f_{ri} - f_{rj})^2 S_{ij} \\
= & \sum_{ij} \left(f_{ri}^2 + f_{rj}^2 - 2 f_{ri} f_{rj}\right) S_{ij} \\
= & 2 \sum_{ij} f_{ri}^2 S_{ij} - 2 \sum_{ij} f_{ri} S_{ij} f_{rj} \\
= & 2\mathbf{f}_r^T D \mathbf{f}_r - 2\mathbf{f}_r^T S \mathbf{f}_r \\
= & 2\mathbf{f}_r^T L \mathbf{f}_r
\end{aligned}$$

By maximizing $Var(\mathbf{f}_r)$, we prefer those features with large variance which have more representative power. Recall that the variance of a random variable a can be written as follows:

$$Var(a) = \int_{\mathcal{M}} (a-\mu)^2 dP(a)$$

$$\mu = \int_{\mathcal{M}} a dP(a)$$

where $\mathcal{M}$ is the data manifold, μ is the expected value of a and dP is the probability measure. By spectral graph theory [17], dP can be estimated by the diagonal matrix D on the sample points. Thus, the *weighted* data variance can be estimated as follows:

$$Var(\mathbf{f}_r) = \sum_i (\mathbf{f}_{ri} - \mu_r)^2 D_{ii} \tag{6.8}$$

$$\begin{aligned} \mu_r &= \sum_i \left(\mathbf{f}_{ri} \frac{D_{ii}}{\sum_i D_{ii}} \right) \\ &= \frac{1}{(\sum_i D_{ii})} \left(\sum_i \mathbf{f}_{ri} D_{ii} \right) \\ &= \frac{\mathbf{f}_r^T D \mathbf{1}}{\mathbf{1}^T D \mathbf{1}} \end{aligned}$$

To remove the mean from the samples, we define:

$$\tilde{\mathbf{f}}_r = \mathbf{f}_r - \frac{\mathbf{f}_r^T D \mathbf{1}}{\mathbf{1}^T D \mathbf{1}} \mathbf{1}$$

Thus,

$$Var(\mathbf{f}_r) = \sum_i \tilde{\mathbf{f}}_{ri}^2 D_{ii} = \tilde{\mathbf{f}}_r^T D \tilde{\mathbf{f}}_r$$

Also, it is easy to show that $\tilde{\mathbf{f}}_r^T L \tilde{\mathbf{f}}_r = \mathbf{f}_r^T L \mathbf{f}_r$ (please see Proposition 3 in Section 6.3.2 for details). We finally get equation (6.6).

It would be important to note that, if we do not remove the mean, the vector $\mathbf{f}_r$ can be a non-zero constant vector such as $\mathbf{1}$. It is easy to check that, $\mathbf{1}^T L \mathbf{1} = 0$ and $\mathbf{1}^T D \mathbf{1} > 0$. Thus, $L_r = 0$. Unfortunately, this feature is clearly of no use since it contains no information. With mean being removed, the new vector $\tilde{\mathbf{f}}_r$ is orthogonal to $\mathbf{1}$ with respect to D, i.e. $\tilde{\mathbf{f}}_r^T D\mathbf{1} = 0$. Therefore, $\tilde{\mathbf{f}}_r$ can not be any constant vector other than $\mathbf{0}$. If $\tilde{\mathbf{f}}_r = \mathbf{0}$, $\tilde{\mathbf{f}}_r^T L\tilde{\mathbf{f}}_r = \tilde{\mathbf{f}}_r^T D\tilde{\mathbf{f}}_r = 0$. Thus, the Laplacian Score L_r becomes a trivial solution and the r-th feature is excluded from selection. While computing the weighted variance, the matrix D models the importance (or local density) of the data points. We can also simply replace it by the identity matrix I, in which case the weighted variance becomes the standard variance. To be specific,

$$\tilde{\mathbf{f}}_r = \mathbf{f}_r - \frac{\mathbf{f}_r^T I\mathbf{1}}{\mathbf{1}^T I\mathbf{1}}\mathbf{1} = \mathbf{f}_r - \frac{\mathbf{f}_r^T\mathbf{1}}{n}\mathbf{1} = \mathbf{f}_r - \mu\mathbf{1}$$

where μ is the mean of $f_{ri}, i = 1, \cdots, n$. Thus,

$$Var(\mathbf{f}_r) = \tilde{\mathbf{f}}_r^T I\tilde{\mathbf{f}}_r = \frac{1}{n}\left(\mathbf{f}_r - \mu\mathbf{1}\right)^T\left(\mathbf{f}_r - \mu\mathbf{1}\right) \tag{6.9}$$

which is just the *standard* variance.

In fact, the Laplacian scores can be thought of as the Rayleigh quotients for the features with respect to the graph G, please see [17] for details.

6.3.2 Connections to Fisher Criterion Score

In this section, we provide a theoretical analysis of the connection between our algorithm and the canonical Fisher criterion score.

Given a set of data points with label, $\{\mathbf{x}_i, y_i\}_{i=1}^n$, $y_i \in \{1, \cdots, c\}$. Let n_i denote the number of data points in class i and μ_i the mean of class i, $i = 1, \cdots, c$. Let μ denote the mean of the whole data set. Naturally, we can build a graph with weight matrix as follows:

$$S_{ij} = \begin{cases} \frac{1}{n_l}, & y_i = y_j = l; \\ 0, & \text{otherwise.} \end{cases} \tag{6.10}$$

Without loss of generality, we assume that the data points are ordered according to which class they are in, so that $\{\mathbf{x}_1, \cdots, \mathbf{x}_{n_1}\}$ are in the first class, $\{\mathbf{x}_{n_1+1}, \cdots, \mathbf{x}_{n_1+n_2}\}$

are in the second class, etc. Thus, S can be written as follows:

$$S = \begin{pmatrix} S_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & S_c \end{pmatrix}$$

where $S_i = \frac{1}{n_i}\mathbf{1}\mathbf{1}^T$ is an $n_i \times n_i$ matrix. For each S_i, the raw (or column) sum is equal to 1, so $D_i = diag(S_i\mathbf{1})$ is just the identity matrix. Define $\mathbf{f}_r^1 = [f_{r1}, \cdots, f_{rn_1}]^T$, $\mathbf{f}_r^2 = [f_{r,n_1+1}, \cdots, f_{r,n_1+n_2}]^T$, etc. Let σ_i^2 denote the variance of the i-th class, corresponding to the r-th feature. Let σ^2 denote the variance of the whole data set.

Proposition 3. *With the weight matrix S defined in (6.10), we have $\tilde{\boldsymbol{f}}_r^T L\tilde{\boldsymbol{f}}_r = \boldsymbol{f}_r^T L\boldsymbol{f}_r = \sum_i n_i\sigma_i^2$, where $L = D - S$.*

Proof. Define $L_i = D_i - S_i = I_i - S_i$, where I_i is the $n_i \times n_i$ identity matrix. We have

$$\begin{aligned} \mathbf{f}_r^T L\mathbf{f}_r &= \sum_{i=1}^{c}(\mathbf{f}_r^i)^T L_i \mathbf{f}_r^i = \sum_{i=1}^{c}(\mathbf{f}_r^i)^T(I_i - \frac{1}{n_i}\mathbf{1}\mathbf{1}^T)\mathbf{f}_r^i \\ &= \sum_{i=1}^{c} n_i cov(\mathbf{f}_r^i, \mathbf{f}_r^i) = \sum_{i=1}^{c} n_i\sigma_i^2 \end{aligned}$$

Note that, since $\mathbf{u}^T L\mathbf{1} = \mathbf{1}^T L\mathbf{u} = 0$, $\forall \mathbf{u} \in \mathbf{R}^n$, the value of $\mathbf{f}_r^T L\mathbf{f}_r$ remains unchanged by subtracting a constant vector ($= \alpha\mathbf{1}$) from $\mathbf{f}_r$. This shows that $\tilde{\mathbf{f}}_r^T L\tilde{\mathbf{f}}_r = \mathbf{f}_r^T L\mathbf{f}_r = \sum_i n_i\sigma_i^2$. □ □

Proposition 4. *With the weight matrix S defined in (6.10), we have $\tilde{\boldsymbol{f}}_r^T D\tilde{\boldsymbol{f}}_r = n\sigma^2$.*

Proof. By the definition of S, we have $D = I$. Thus, this is a immediate result from equation (6.9). □ □

Proposition 5. *With the weight matrix S defined in (6.10), we have $\sum_{i=1}^{c} n_i(\mu_i - \mu)^2 = \tilde{\boldsymbol{f}}_r^T D\tilde{\boldsymbol{f}}_r - \tilde{\boldsymbol{f}}_r^T L\tilde{\boldsymbol{f}}_r$.*

Proof.

$$\begin{aligned}
& \sum_{i=1}^{c} n_i(\mu_i - \mu)^2 \\
= & \sum_{i=1}^{c} \left(n_i\mu_i^2 - 2n_i\mu_i\mu + n_i\mu^2\right) \\
= & \sum_{i=1}^{c} \frac{1}{n_i}(n_i\mu_i)^2 - 2\mu\sum_{i=1}^{c} n_i\mu_i + \mu^2\sum_{i=1}^{c} n_i \\
= & \sum_{i=1}^{c} \frac{1}{n_i}\left((\mathbf{f}_r^i)^T\mathbf{1}\mathbf{1}^T\mathbf{f}_r^i\right) - 2n\mu^2 + n\mu^2 \\
= & \sum_{i=1}^{c} \mathbf{f}_r^i S_i \mathbf{f}_r^i - \frac{1}{n}(n\mu)^2 \\
= & \mathbf{f}_r^T S\mathbf{f}_r - \mathbf{f}_r^T(\frac{1}{n}\mathbf{1}\mathbf{1}^T)\mathbf{f}_r \\
= & \mathbf{f}_r^T(I - S)\mathbf{f}_r - \mathbf{f}_r^T(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)\mathbf{f}_r \\
= & \mathbf{f}_r^T L\mathbf{f}_r - n\sigma^2 \\
= & \tilde{\mathbf{f}}_r^T L\tilde{\mathbf{f}}_r^T - \tilde{\mathbf{f}}_r^T D\tilde{\mathbf{f}}_r^T
\end{aligned}$$

This completes the proof. □ □

We finally get the following theorem.

Theorem 1. *Let F_r denote the Fisher Criterion score of the r-th feature. With the weight matrix S defined in (6.10), we have $L_r = \frac{1}{1+F_r}$.*

Proof. From propositions 3,4,5, we see that

$$\begin{aligned}
F_r &= \frac{\sum_{i=1}^{c} n_i(\mu_i - \mu)^2}{\sum_{i=1}^{c} n_i\sigma_i^2} \\
&= \frac{\tilde{\mathbf{f}}_r^T D\tilde{\mathbf{f}}_r^T - \tilde{\mathbf{f}}_r^T L\tilde{\mathbf{f}}_r^T}{\tilde{\mathbf{f}}_r^T L\tilde{\mathbf{f}}_r^T} \\
&= \frac{1}{L_r} - 1
\end{aligned}$$

Thus, $L_r = \frac{1}{1+F_r}$. □ □

Theorem 1 tells us that, when the weight matrix is defined as in (6.10), our algorithm gives the same result as Fisher criterion score. In unsupervised learning scenario, the label information is unavailable. However, in many cases, two data points probably share the same label in they are close to each other. Therefore, the nearest neighbor graph defined in our algorithm can give a optimal unsupervised approximation to the weighted graph defined in (6.10). This shows that our algorithm is capable of finding those features with most discriminative power, even though the label information is unavailable.

6.4 Experimental Results

Several experiments were carried out to demonstrate the efficiency and effectiveness of our algorithm. Our algorithm is a unsupervised filter method, while almost all the existing filter methods are supervised. Therefore, we compared our algorithm with data variance which can be performed in unsupervised manner.

6.4.1 UCI Iris Data

Iris dataset, popularly used for testing clustering and classification algorithms, is taken from UCI ML repository [14]. It contains 3 classes of 50 instances each, where each class refers to a type of Iris plant. Each instance is characterized by four features, i.e. sepal length, sepal width, petal length, and petal width. One class is linearly separable from the other two, but the other two are not linearly separable from each other. Out of the four features it is known that the features F3 (petal length) and F4 (petal width) are more important for the underlying clusters.

The class correlation for each feature is 0.7826, -0.4194, 0.9490 and 0.9565. We also used leave-one-out strategy to do classification by using each single feature. We simply used the nearest neighbor classifier. The classification error rates for the four features are 0.41, 0.52, 0.12 and 0.12, respectively. Our analysis indicates that F3 and F4 are better than F1 and F2 in the sense of discrimination. In figure 6.1, we present a 2-D visualization of the Iris data.

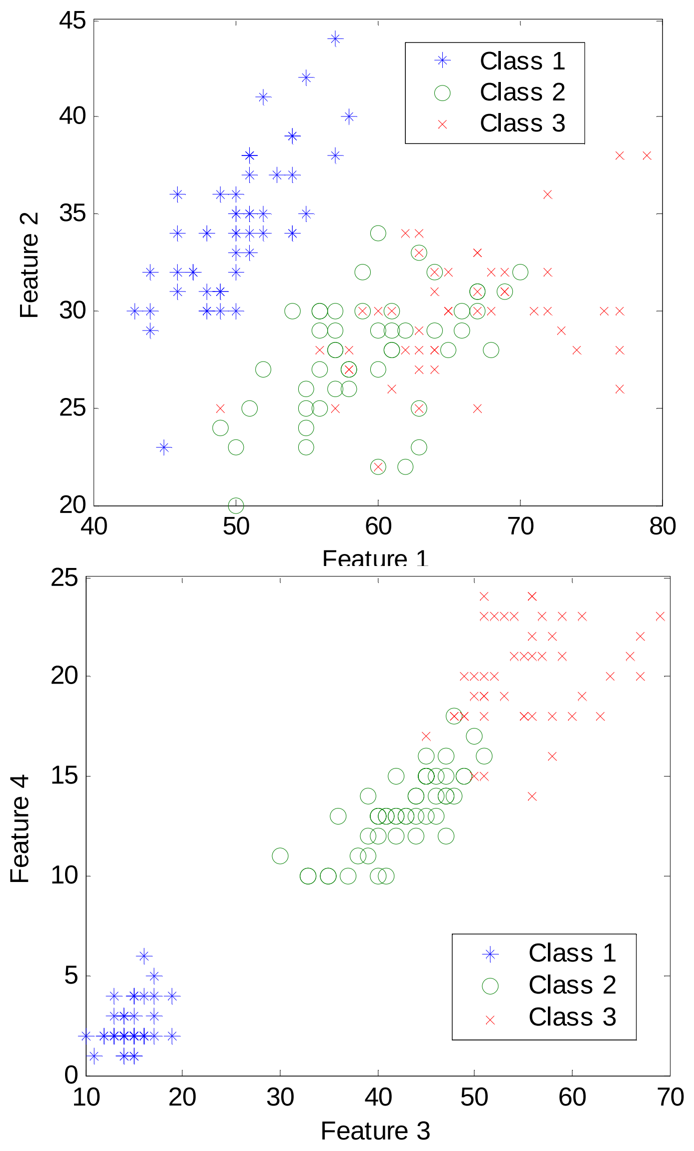


Figure 6.1: 2-D visualization of the Iris data.

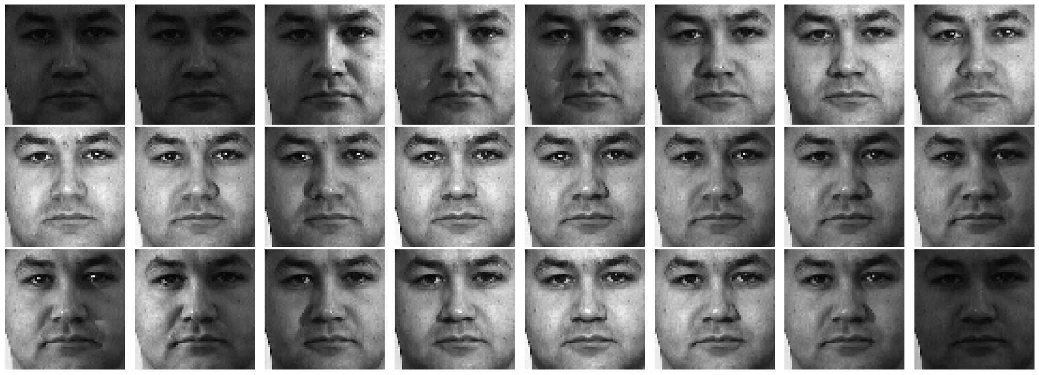


Figure 6.2: Sample face images from the CMU PIE face database. For each subject, there are 24 face images under different lighting conditions with fixed pose and expression.

We compared three methods, i.e. Variance, Fisher Criterion Score and Laplacian Score for feature selection. All of them are filter methods which are independent to any learning tasks. However, Fisher Criterion Score is supervised, while the other two are unsupervised.

By using variance, the four features are sorted as F3, F1, F4, F2. Laplacian score (with $k \geq 15$) sorts these four features as F3, F4, F1, F2. Laplacian score (with $3 \leq k < 15$) sorts these four features as F4, F3, F1, F2. With a larger k, we see more global structure of the data set. Therefore, the feature F3 is ranked above F4 since the variance of F3 is greater than that of F4. By using Fisher criterion score, the four features are sorted as F3, F4, F1, F2. This indicates that Laplacian score (unsupervised) achieved the same result as Fisher criterion score (supervised).

6.4.2 Face Clustering on PIE

In this section, we apply our feature selection algorithm to face clustering. Our approach is based on appearance model [36] [49]. The geometric concept that is central to appearance-based methods is the idea of appearance manifold introduced in [55]. By using Laplacian score, we select a subset of features which are the most useful for discrimination. Clustering is then performed in such a subspace. We expect that clustering in subspace can obtain better performance than that in the original space.

Data Preparation

The CMU PIE face database [69] is used in this experiment. It contains 68 subjects with 41,368 face images as a whole. The face images were captured under varying pose, illumination and expression. Preprocessing to locate the faces was applied. Original images were normalized (in scale and orientation) such that the two eyes were aligned at the same position. Then, the facial areas were cropped into the final images for matching. The size of each cropped image is 32×32 pixels, with 256 grey levels per pixel. Thus, each image is represented by a 1024-dimensional vector. No further preprocessing is done. In this experiment, we fixed the pose and expression. Thus, for each subject, we got 24 images under different lighting conditions. Figure 6.2 shows some sample images for a certain subject.

For each given number k, k classes were randomly selected from the face database. This process was repeated 20 times (except for $k = 68$) and the average performance was computed. For each single test (given k classes of faces), two algorithms, i.e. feature selection using variance and Laplacian score are used to select the features. The K-means was then performed in the selected feature subspace. Again, the K-means was repeated 10 times with different initializations and the best result in terms of the objective function of K-means was recorded.

Evaluation Metrics

The clustering result is evaluated by comparing the obtained label of each data point with that provided by the data corpus. Two metrics, the accuracy (AC) and the normalized mutual information metric ($\overline{MI}$) are used to measure the clustering performance [81]. Given a data point $\mathbf{x}_i$, let r_i and s_i be the obtained cluster label and the label provided by the data corpus, respectively. The AC is defined as follows:

$$AC = \frac{\sum_{i=1}^{n} \delta(s_i, map(r_i))}{n} \tag{6.11}$$

Table 6.1: Accuracy comparisons (k is the number of clusters)

k	Feature Number	20	50	100	200	300	500	1024
5	Laplacian Score	0.727	0.806	0.831	0.849	0.837	0.644	0.479
	Variance	0.683	0.698	0.602	0.503	0.482	0.464	0.479
10	Laplacian Score	0.685	0.743	0.787	0.772	0.711	0.585	0.403
	Variance	0.494	0.500	0.456	0.418	0.392	0.392	0.403
30	Laplacian Score	0.591	0.623	0.671	0.650	0.588	0.485	0.358
	Variance	0.399	0.393	0.390	0.365	0.346	0.340	0.358
68	Laplacian Score	0.479	0.554	0.587	0.608	0.553	0.465	0.332
	Variance	0.328	0.362	0.334	0.316	0.311	0.312	0.332

where n is the total number of data points and $\delta(x, y)$ is the delta function that equals one if $x = y$ and equals zero otherwise, and $\text{map}(r_i)$ is the permutation mapping function that maps each cluster label r_i to the equivalent label from the data corpus. The best mapping can be found by using the Kuhn-Munkres algorithm [51].

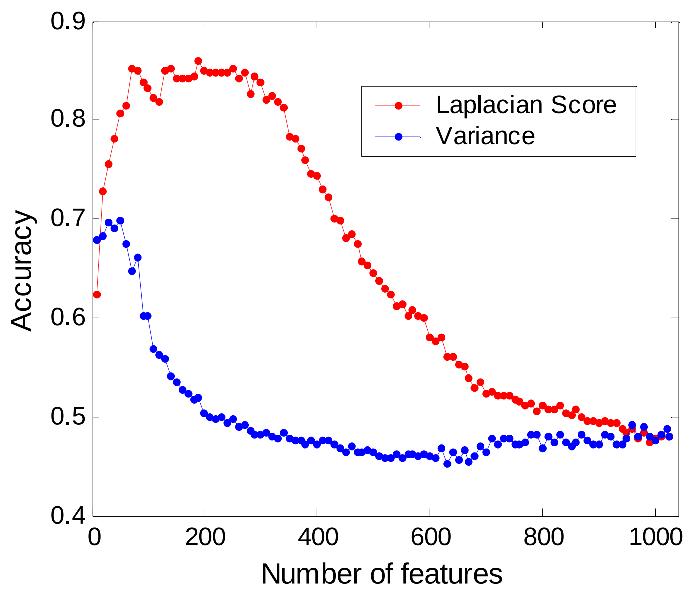
Let C denote the set of clusters obtained from the ground truth and C' obtained from our algorithm. Their mutual information metric $MI(C, C')$ is defined as follows:

$$MI(C, C') = \sum_{c_i \in C, c'_j \in C'} p(c_i, c'_j) \cdot log_2 \frac{p(c_i, c'_j)}{p(c_i) \cdot p(c'_j)} \tag{6.12}$$

where $p(c_i)$ and $p(c'_j)$ are the probabilities that a data point arbitrarily selected from the corpus belongs to the clusters c_i and c'_j, respectively, and $p(c_i, c'_j)$ is the joint probability that the arbitrarily selected data point belongs to the clusters c_i as well as c'_j at the same time. In our experiments, we use the normalized mutual information $\overline{MI}$ as follows:

$$\overline{MI}(C, C') = \frac{MI(C, C')}{\max(H(C), H(C'))} \tag{6.13}$$

where $H(C)$ and $H(C')$ are the entropies of C and C', respectively. It is easy to check that $\overline{MI}(C, C')$ ranges from 0 to 1. $\overline{MI} = 1$ if the two sets of clusters are identical, and $\overline{MI} = 0$ if the two sets are independent.



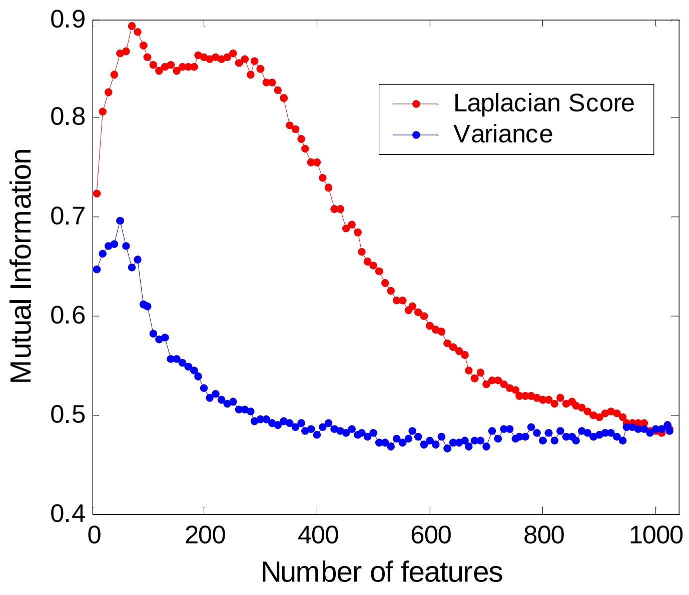
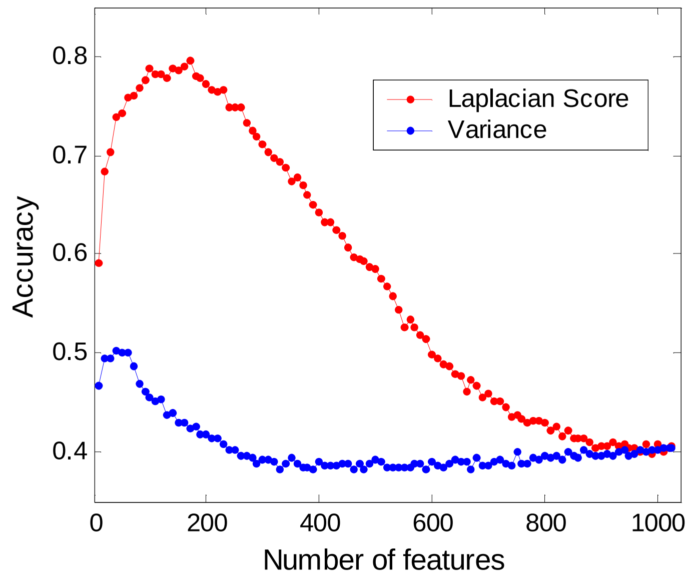


Figure 6.3: Clustering results on 5 randomly chosen classes.



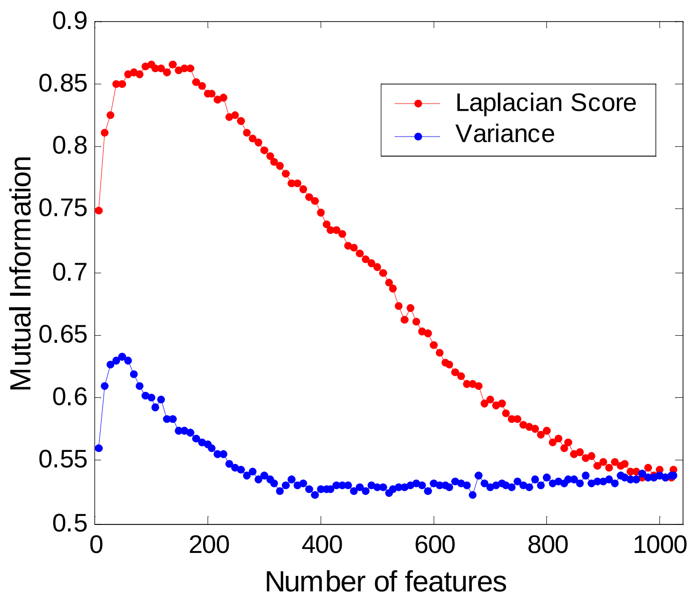
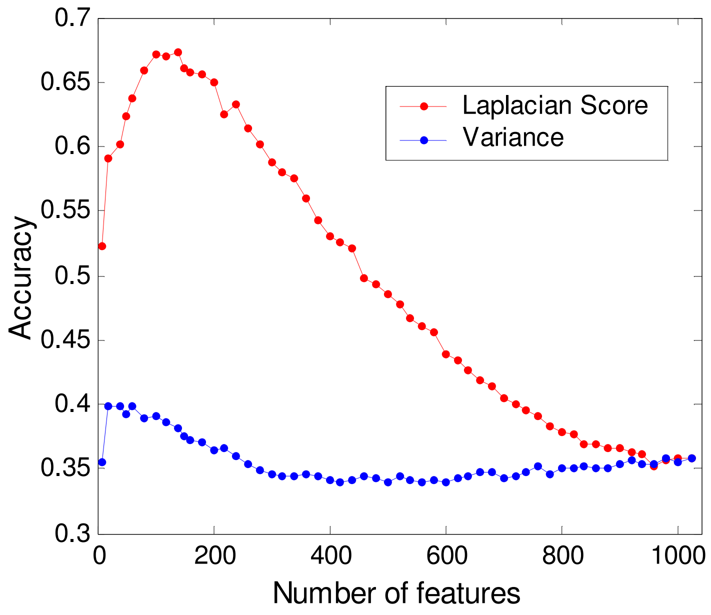


Figure 6.4: Clustering results on 10 randomly chosen classes.



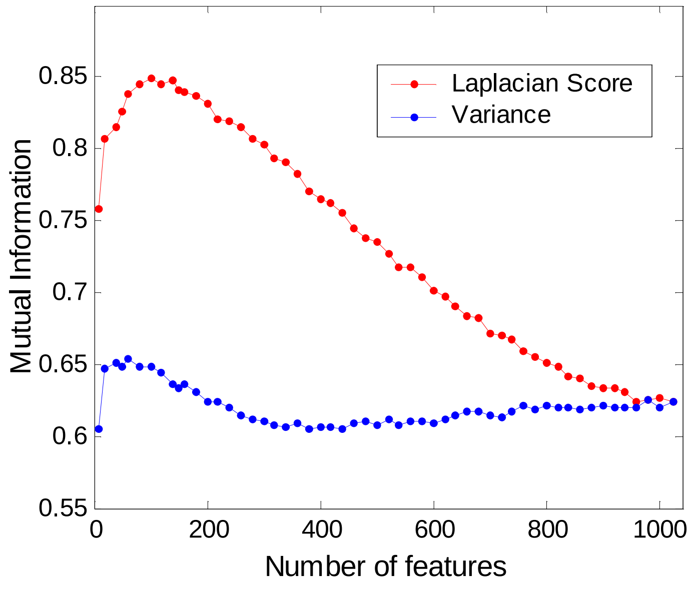
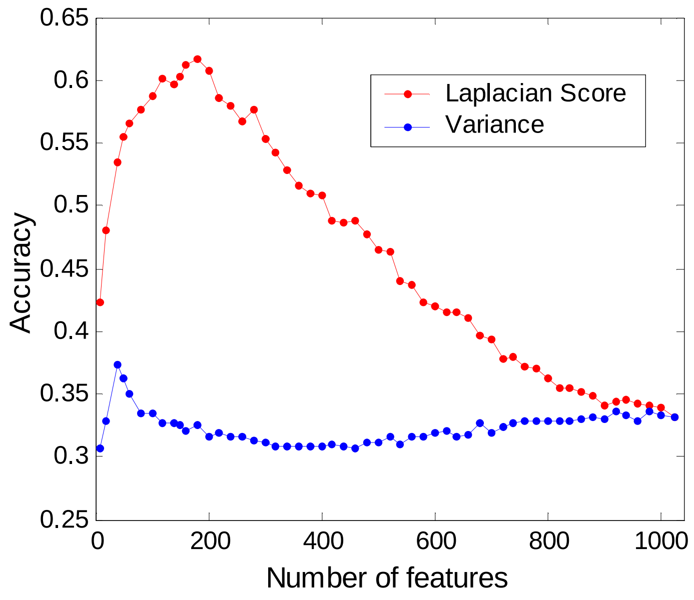


Figure 6.5: Clustering results on 30 randomly chosen classes.



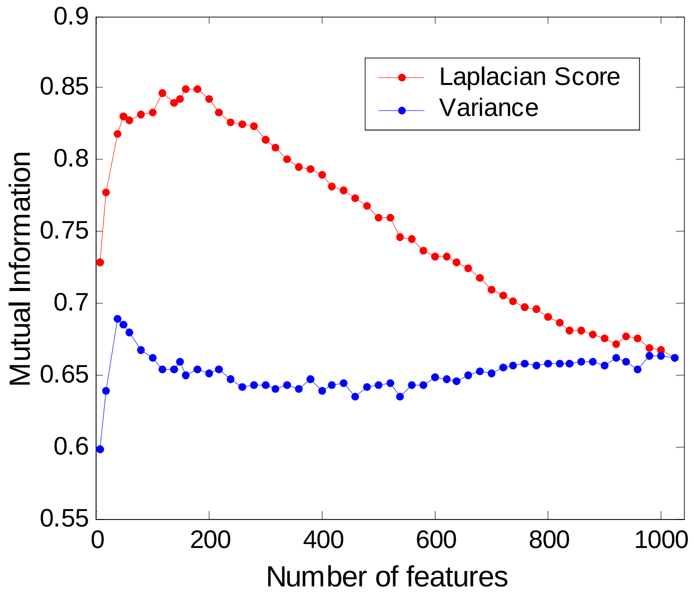


Figure 6.6: Clustering results on 68 classes.

Table 6.2: Mutual Information comparisons (k is the number of clusters)

k	Feature Number	20	50	100	200	300	500	1024
5	Laplacian Score	0.807	0.866	0.861	0.862	0.85	0.652	0.484
	Variance	0.662	0.697	0.609	0.526	0.495	0.482	0.484
10	Laplacian Score	0.811	0.849	0.865	0.842	0.796	0.705	0.538
	Variance	0.609	0.632	0.6	0.563	0.538	0.529	0.538
30	Laplacian Score	0.807	0.826	0.849	0.831	0.803	0.735	0.624
	Variance	0.646	0.649	0.649	0.624	0.611	0.608	0.624
68	Laplacian Score	0.778	0.83	0.833	0.843	0.814	0.76	0.662
	Variance	0.639	0.686	0.661	0.651	0.642	0.643	0.662

Results

We tested our algorithm with different numbers of clusters (k=5, 10, 30, 68). In all the tests, the number of nearest neighbors in our algorithm is taken to be 5. The experimental results are shown in Figure 3-6 and Table 1,2. As can be seen, in all these cases, our algorithm performs much better than using variance for feature selection. The clustering performance varies with the number of features. The best performance is obtained at very low dimensionality (less than 200). This indicates that feature selection is capable of enhancing clustering performance.

In the above experiments, all the data are unlabelled. However, in some situations, the data set might be partially labelled. Thus, we can incorporate the label information into the graph structure by putting an edge between two nodes if they share the same label. This is left for our future work.

CHAPTER 7
FROM VECTOR TO TENSOR — TENSOR SUBSPACE ANALYSIS

Traditionally, an image (face image in particular) of size $n_1 \times n_2$ is considered as a point of the vector space $\mathbb{R}^{n_1 \times n_2}$. An image vector is formed by concatenating all the column vectors (or row vectors) of the image matrix together. The necessity of such "matrix-to-vector" conversion is due to the fact that most of the current learning algorithms can only be applied to the vectorial data.

Recently, multilinear algebra, the algebra of higher-order tensors, was applied for analyzing the multifactor structure of image ensembles [76], [84], [39]. Vasilescu and Terzopoulos have proposed a novel face representation algorithm called Tensorface [76]. Tensorface represents the set of face images by a higher-order tensor and extends Singular Value Decomposition (SVD) to higher-order tensor data. In this way, the multiple factors related to expression, illumination and pose can be separated from different dimensions of the tensor.

In this Chapter, we introduce a new algorithm for image (human faces in particular) representation based on the considerations of multilinear algebra and differential geometry. We call it *Tensor Subspace Analysis* (TSA). For an image of size $n_1 \times n_2$, it is represented as the second order tensor (or, matrix) in the tensor space $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$. On the other hand, the face space is generally a submanifold embedded in $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$. Given some images sampled from the face manifold, we can build an adjacency graph to model the local geometrical structure of the manifold. TSA finds a projection that respects this graph structure. The obtained tensor subspace provides an optimal linear approximation to the face manifold in the sense of local isometry. Vasilescu shows how to extend SVD(PCA) to higher order tensor data. We extend Laplacian based idea to tensor data.

It is worthwhile to highlight several aspects of the proposed approach here:

1. While traditional linear dimensionality reduction algorithms like PCA, LDA and LPP find a map from $\mathbb{R}^n$ to $\mathbb{R}^l$ $(l < n)$, TSA finds a map from $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$ to $\mathcal{R}^{l_1} \otimes \mathcal{R}^{l_2}$ $(l_1 < n_1, l_2 < n_2)$. This leads to structured dimensionality reduction.

2. TSA can be performed in either supervised, unsupervised, or semi-supervised manner. When label information is available, it can be easily incorporated into the graph structure. Also, by preserving neighborhood structure, TSA is less sensitive to noise and outliers.

3. The computation of TSA is very simple. It can be obtained by solving two eigenvector problems. The matrices in the eigen-problems are of size $n_1 \times n_1$ or $n_2 \times n_2$, which are much smaller than the matrices of size $n \times n$ ($n = n_1 \times n_2$) in PCA, LDA and LPP. Therefore, TSA is much more computationally efficient in time and storage. There are few parameters that are independently estimated, so performance in small data sets is very good.

4. TSA explicitly takes into account the manifold structure of the image space. The local geometrical structure is modeled by an adjacency graph.

5. This paper is primarily focused on the second order tensors (or, matrices). However, the algorithm and analysis presented here can also be applied to higher order tensors.

7.1 The Algebra of Tensor

In this section, we provide a brief overview of the algebra of tensors. For a detailed treatment please see [46].

A tensor with order k is a real-valued multilinear function on k vector spaces:

$$T : \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_k} \to \mathbb{R}$$

The number k is called the *order* of T. A multilinear function is linear as a function of each variable separately. The set of all k-tensors on $\mathbb{R}^{n_i}, i = 1, \cdots, k$, denoted by $\mathcal{T}^k$, is a vector space under the usual operations of pointwise addition and scalar multiplication:

$$(aT)(\mathbf{a}_1, \cdots, \mathbf{a}_k) = a\left(T(\mathbf{a}_1, \cdots, \mathbf{a}_k)\right),$$

$$(T + T')(\mathbf{a}_1, \cdots, \mathbf{a}_k) = T(\mathbf{a}_1, \cdots, \mathbf{a}_k) + T'(\mathbf{a}_1, \cdots, \mathbf{a}_k)$$

where $\mathbf{a}_i \in \mathbb{R}^{n_i}$. Given two tensors $S \in \mathcal{T}^k$ and $T \in \mathcal{T}^l$, define a map:

$$S \otimes T : \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_{k+l}} \to \mathbb{R}$$

by

$$S \otimes T(\mathbf{a}_1, \cdots, \mathbf{a}_{k+l}) = S(\mathbf{a}_1, \cdots, \mathbf{a}_k) T(\mathbf{a}_{k+1}, \cdots, \mathbf{a}_{k+l})$$

It is immediate from the multilinearity of S and T that $S \otimes T$ depends linearly on each argument $\mathbf{a}_i$ separately, so it is a $(k+l)$-*tensor*, called the tensor product of S and T.

For the first order tensors, they are simply the covectors on $\mathbb{R}^{n_1}$. That is, $\mathcal{T}^1 = \mathcal{R}^{n_1}$, where $\mathcal{R}^{n_1}$ is the dual space of $\mathbb{R}^{n_1}$. The second order tensor space is a product of two first order tensor spaces, i.e. $\mathcal{T}^2 = \mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$. Let $\mathbf{e}_1, \cdots, \mathbf{e}_{n_1}$ be the standard basis of $\mathbb{R}^{n_1}$, and $\varepsilon_1, \cdots, \varepsilon_{n_1}$ be the dual basis of $\mathcal{R}^{n_1}$ which is formed by coordinate functions with respect to the basis of $\mathbb{R}^{n_1}$. Likewise, let $\widetilde{\mathbf{e}}_1, \cdots, \widetilde{\mathbf{e}}_{n_2}$ be a basis of $\mathbb{R}^{n_2}$, and $\widetilde{\varepsilon}_1, \cdots, \widetilde{\varepsilon}_{n_2}$ be the dual basis of $\mathcal{R}^{n_2}$. We have,

$$\varepsilon_i(\mathbf{e}_j) = \delta_{ij} \text{ and } \widetilde{\varepsilon}_i(\widetilde{\mathbf{e}}_j) = \delta_{ij}$$

where δ_{ij} is the kronecker delta function. Thus, $\{\varepsilon_i \otimes \widetilde{\varepsilon}_j\}$ $(1 \leq i \leq n_1,\ 1 \leq j \leq n_2)$ forms a basis of $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$. For any 2-tensor T, we can write it as:

$$T = \sum_{\substack{1 \leq i \leq n_1 \\ 1 \leq j \leq n_2}} T_{ij} \varepsilon_i \otimes \widetilde{\varepsilon}_j$$

This shows that every 2-tensor in $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$ uniquely corresponds to a $n_1 \times n_2$ matrix.

Given two vectors $\mathbf{a} = \sum_{k=1}^{n_1} a_k \mathbf{e}_k \in \mathbb{R}^{n_1}$ and $\mathbf{b} = \sum_{l=1}^{n_2} b_l \widetilde{\mathbf{e}}_l \in \mathbb{R}^{n_2}$, we have

$$\begin{aligned}
T(\mathbf{a}, \mathbf{b}) &= \sum_{ij} T_{ij} \varepsilon_i \otimes \widetilde{\varepsilon}_j (\sum_{k=1}^{n_1} a_k \mathbf{e}_k, \sum_{l=1}^{n_2} b_l \widetilde{\mathbf{e}}_l) \\
&= \sum_{ij} T_{ij} \varepsilon_i (\sum_{k=1}^{n_1} a_k \mathbf{e}_k) \widetilde{\varepsilon}_j (\sum_{l=1}^{n_2} b_l \widetilde{\mathbf{e}}_l) \\
&= \sum_{ij} T_{ij} a_i b_j \\
&= \mathbf{a}^T T \mathbf{b}
\end{aligned}$$

Note that, in this paper our primary interest is focused on the second order tensors. However, the algebra presented here and the algorithm presented in the next section can also be applied to higher order tensors.

7.2 Tensor Subspace Analysis

In this section, we introduce a new algorithm called *Tensor Subspace Analysis* for learning a tensor subspace which respects the geometrical and discriminative structures of the original data space.

7.2.1 Laplacian based Dimensionality Reduction

Problems of dimensionality reduction has been considered. One general approach is based on graph Laplacian [10]. The objective function of Laplacian eigenmap is as follows:

$$\min_f \sum_{ij} \left(f(\mathbf{x}_i) - f(\mathbf{x}_j)\right)^2 S_{ij}$$

where S is a similarity matrix. These optimal functions are nonlinear but may be expensive to compute.

A class of algorithms may be optimized by restricting problem to more tractable families of functions. One natural approach restricts to linear function giving rise to LPP [33]. In this paper we will consider a more structured subset of linear functions that arise out of tensor analysis. This provided greater computational benefits.

7.2.2 The Linear Dimensionality Reduction Problem in Tensor Space

The generic problem of linear dimensionality reduction in the second order tensor space is the following. Given a set of data points $X_1, \cdots, X_m$ in $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$, find two transformation matrices U of size $n_1 \times l_1$ and V of size $n_2 \times l_2$ that maps these m points to a set of points $Y_1, \cdots, Y_m \in \mathcal{R}^{l_1} \otimes \mathcal{R}^{l_2} (l_1 < n_1, l_2 < n_2)$, such that Y_i "represents" X_i, where $Y_i = U^T X_i V$. Our method is of particular applicability in the special case where $X_1, \cdots, X_m \in \mathcal{M}$ and $\mathcal{M}$ is a nonlinear submanifold embedded in $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$.

7.2.3 Coordinate System Transformation

Let $\{\mathbf{u}_k\}_{k=1}^{n_1}$ be the new orthonormal basis of $\mathcal{R}^{n_1}$ and $\{\mathbf{v}_l\}_{l=1}^{n_2}$ be the new orthonormal basis of $\mathcal{R}^{n_2}$. We have

$$\mathbf{u}_k = (u_{k1}, \cdots, u_{kn_1})^T = \sum_i u_{ki} \varepsilon_i$$

and

$$\mathbf{v}_l = (v_{l1}, \cdots, v_{ln_2})^T = \sum_j v_{lj} \widetilde{\varepsilon}_j$$

It is easy to show that:

$$\varepsilon_i = \sum_{k=1}^{n_1} u_{ki} \mathbf{u}_k$$

$$\widetilde{\varepsilon}_j = \sum_{l=1}^{n_2} v_{lj} \mathbf{v}_l$$

Thus, an image tensor can be computed using the new basis as follows:

$$\begin{aligned}
T &= \sum_{ij} T_{ij} \varepsilon_i \otimes \widetilde{\varepsilon}_j \\
&= \sum_{ij} T_{ij} \left(\sum_{k=1}^{n_1} u_{ki} \mathbf{u}_k \right) \otimes \left(\sum_{l=1}^{n_2} v_{lj} \mathbf{v}_l \right) \\
&= \sum_{kl} \left(\sum_{ij} T_{ij} u_{ki} v_{lj} \right) \mathbf{u}_k \otimes \mathbf{v}_l \\
&= \sum_{kl} \left(\mathbf{u}_k^T T \mathbf{v}_l \right) \mathbf{u}_k \otimes \mathbf{v}_l \qquad (7.1)
\end{aligned}$$

Eqn. 7.1 shows that $\{\mathbf{u}_i \otimes \mathbf{v}_j\}$ forms a new basis of the tensor space $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$. Thus, the tensor T becomes $U^T T V$ with respect to the new basis, where $U = (\mathbf{u}_1, \cdots, \mathbf{u}_{n_1})$ and $V = (\mathbf{v}_1, \cdots, \mathbf{v}_{n_2})$. Specifically, the projection of T on the basis $\mathbf{u}_i \otimes \mathbf{v}_j$ can be computed as their inner product:

$$< T, \mathbf{u}_i \otimes \mathbf{v}_j > = < T, \mathbf{u}_i \mathbf{v}_j^T > = \mathbf{u}_i^T T \mathbf{v}_j$$

In this paper, we consider an image as a 2-tensor. For the basis tensors $\mathbf{u}_i \otimes \mathbf{v}_j$, or $\mathbf{u}_i \mathbf{v}_j^T$ in matrix form, they can also be considered as images. We call them *TSAs*. Therefore, for any image, it can be represented as a linear combination of the *TSAs*.

7.2.4 Optimal Linear Embedding

As we described previously, the images are probably sampled from a low dimensional submanifold embedded in the ambient space. One hopes then to estimate geometrical and topological properties of the submanifold from random points ("scattered data") lying on this unknown submanifold. Particularly, We consider the problem of finding a linear subspace approximation to the submanifold in the sense of local isometry.

Let X denote the second order image tensor, or matrix. Given m data points $\mathcal{X} = \{X_1, \cdots, X_m\}$ sampled from the face submanifold $\mathcal{M} \in \mathcal{R}^{n_1} \otimes \mathcal{R}^{n_1}$, one can build a nearest neighbor graph $\mathcal{G}$ to model the local geometrical structure of $\mathcal{M}$. Let

S be the weight matrix of $\mathcal{G}$. A possible definition of S is as follows:

$$S_{ij} = \begin{cases} 1, & \text{if } X_i \text{ is among the } p \text{ nearest} \\ & \text{neighbors of } X_j, \text{ or } X_j \text{ is among} \\ & \text{the } p \text{ nearest neighbors of } X_i; \\ 0, & \text{otherwise.} \end{cases} \tag{7.2}$$

There are also other definitions of S, such as using heat kernel. Please see [10] for details. When the label information is available, it can be easily incorporated into the graph as follows:

$$S_{ij} = \begin{cases} e^{-\frac{\|X_i - X_j\|^2}{t}}, & \text{if } X_i \text{ and } X_j \text{ share the same label;} \\ 0, & \text{otherwise.} \end{cases} \tag{7.3}$$

Let U and V be the transformation matrices. A reasonable transformation respecting the graph structure can be obtained by solving the following objective functions:

$$\min_{U,V} \sum_{ij} \|U^T X_i V - U^T X_j V\|^2 S_{ij} \tag{7.4}$$

The objective function incurs a heavy penalty if neighboring points X_i and X_j are mapped far apart. Therefore, minimizing it is an attempt to ensure that if X_i and X_j are "close" then $U^T X_i V$ and $U^T X_j V$ are "close" as well.

Our objective function in (7.4) can also be interpreted from the perspective of *min-cut* graph partitioning. Without loss of generality, we consider the problem of dividing the graph $\mathcal{G}$ into two disjoint subgraphs, $\mathcal{G}_1$ and $\mathcal{G}_2$, such that $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$ and $\mathcal{G}_1 \cap \mathcal{G}_2 = \emptyset$. The min-cut criteria can be stated as follows:

$$\min_{\mathcal{G}_1,\mathcal{G}_2} \sum_{i\in\mathcal{G}_1} \sum_{j\in\mathcal{G}_2} S_{ij} \tag{7.5}$$

Now, for each node (data point) X_i, we assign it a label y_i to indicate whether it is in $\mathcal{G}_1$ or $\mathcal{G}_2$, $y_i \in \{-1, 1\}$. If X_i and X_j belong to the same subgraph, $y_i - y_j = 0$; otherwise, $|y_i - y_j| = 2$. Thus, the min-cut graph partitioning problem can be reduced to finding the optimal vector $\mathbf{y}(=(y_1,\cdots,y_m)^T)$ which minimizes the following objective function:

$$\min_{\mathbf{y}, y_i\in\{-1,1\}} \sum_{ij}(y_i-y_j)^2 S_{ij} \tag{7.6}$$

Since y_i can only be 1 or -1, finding the optimal $\mathbf{y}$ is time consuming. In order to reduce the computational complexity, one may resort to spectral relaxation method to approximate the optimal solution. Specifically, we can relax the constraint that $y_i \in \{-1,1\}$, by allowing y_i to be any real number. Further, if we restrict the map from X_i to y_i to be linear, i.e. $y_i = \mathbf{u}^T X_i \mathbf{v}$, we get the objective function (7.4). Please see [54], [30] for the details about spectral graph partitioning.

7.2.5 Derivation

Let $Y_i = U^T X_i V$. Let D be a diagonal matrix, $D_{ii} = \sum_j S_{ij}$. Since $\|A\|^2 = tr(AA^T)$, we see that:

$$\begin{aligned}
& \frac{1}{2}\sum_{ij} \|U^T X_i V - U^T X_j V\|^2 S_{ij} \\
= \; & \frac{1}{2}\sum_{ij} tr\left((Y_i - Y_j)(Y_i - Y_j)^T\right) S_{ij} \\
= \; & \frac{1}{2}\sum_{ij} tr\left(Y_iY_i^T + Y_jY_j^T - Y_iY_j^T - Y_jY_i^T\right) S_{ij} \\
= \; & tr\left(\sum_i D_{ii}Y_iY_i^T - \sum_{ij} S_{ij}Y_iY_j^T\right) \\
= \; & tr\left(\sum_i D_{ii}U^TX_iVV^TX_i^TU - \sum_{ij} S_{ij}U^TX_iVV^TX_j^TU\right) \\
= \; & tr\left(U^T\left(\sum_i D_{ii}X_iVV^TX_i^T - \sum_{ij} S_{ij}X_iVV^TX_j^T\right)U\right) \\
\doteq \; & tr\left(U^T\left(D_V - S_V\right)U\right)
\end{aligned}$$

where $D_V = \sum_i D_{ii} X_i V V^T X_i^T$ and $S_V = \sum_{ij} S_{ij} X_i V V^T X_j^T$. "*tr*" denotes the *trace* operator. Similarly, $\|A\|^2 = trace(A^T A)$, so we also have

$$\begin{aligned}
& \frac{1}{2}\sum_{ij} \|U^T X_i V - U^T X_j V\|^2 S_{ij} \\
= \;& \frac{1}{2}\sum_{ij} tr\left((Y_i - Y_j)^T (Y_i - Y_j)\right) S_{ij} \\
= \;& \frac{1}{2}\sum_{ij} tr\left(Y_i^T Y_i + Y_j^T Y_j - Y_i^T Y_j - Y_j^T Y_i\right) S_{ij} \\
= \;& tr\left(\sum_i D_{ii} Y_i^T Y_i - \sum_{ij} S_{ij} Y_i^T Y_j\right) \\
= \;& tr\left(V^T\left(\sum_i D_{ii} X_i^T U U^T X_i - \sum_{ij} S_{ij} X_i^T U U^T X_j\right) V\right) \\
\doteq \;& tr\left(V^T\left(D_U - S_U\right) V\right)
\end{aligned}$$

where $D_U = \sum_i D_{ii} X_i^T U U^T X_i$ and $S_U = \sum_{ij} S_{ij} X_i^T U U^T X_j$. Therefore, we should simultaneously minimize $tr\Big(U^T\big(D_V - S_V\big)U\Big)$ and $tr\left(V^T\left(D_U - S_U\right)V\right)$.

In addition to preserving the graph structure, we also aim at maximizing the global variance on the manifold. Recall that the variance of a random variable x can be written as follows:

$$var(x) = \int_{\mathcal{M}} (x-\mu)^2 dP(x)$$

$$\mu = \int_{\mathcal{M}} x dP(x)$$

where $\mathcal{M}$ is the data manifold, μ is the expected value of x and dP is the probability measure on the manifold. By spectral graph theory [17], dP can be discretely estimated by the diagonal matrix $D(D_{ii} = \sum_j S_{ij})$ on the sample points. Let $Y = U^T X V$ denote the random variable in the tensor subspace and suppose the data points have a zero mean. Thus, the *weighted* variance can be estimated as follows:

$$\begin{aligned}
var(Y) &= \sum_i \|Y_i\|^2 D_{ii} \\
&= \sum_i tr(Y_i^T Y_i) D_{ii} \\
&= \sum_i tr\left(V^T X_i^T U U^T X_i V\right) D_{ii} \\
&= tr\left(V^T \left(\sum_i D_{ii} X_i^T U U^T X_i\right) V\right) \\
&= tr\left(V^T D_U V\right)
\end{aligned}$$

Similarly, $\|Y_i\|^2 = tr(Y_i Y_i^T)$, so we also have:

$$\begin{aligned}
var(Y) &= \sum_i tr(Y_i Y_i^T) D_{ii} \\
&= tr\left(U^T \left(\sum_i D_{ii} X_i V V^T X_i^T\right) U\right) \\
&= tr\left(U^T D_V U\right)
\end{aligned}$$

Finally, we get the following optimization problems:

$$\min_{U,V} \frac{tr\left(U^T \left(D_V - S_V\right) U\right)}{tr\left(U^T D_V U\right)} \tag{7.7}$$

$$\min_{U,V} \frac{tr\left(V^T \left(D_U - S_U\right) V\right)}{tr\left(V^T D_U V\right)} \tag{7.8}$$

The above two minimization problems (7.7) and (7.8) depends on each other, and hence can not be solved independently. In the following, we describe a simple yet effective computational method to solve these two optimization problems.

7.2.6 Computation

In this subsection, we discuss how to solve the optimization problems (7.7) and (7.8). It is easy to see that the optimal U should be the generalized eigenvectors of $(D_V - S_V, D_V)$ and the optimal V should be the generalized eigenvectors of $(D_U - S_U, D_U)$. However, it is difficult to compute the optimal U and V simultaneously since the matrices D_V, S_V, D_U, S_U are not fixed. In this paper, we compute U and V iteratively as follows. We first fix U, then V can be computed by solving the following generalized eigenvector problem:

$$(D_U - S_U)\mathbf{v} = \lambda D_U \mathbf{v} \tag{7.9}$$

Once V is obtained, U can be updated by solving the following generalized eigenvector problem:

$$(D_V - S_V)\mathbf{u} = \lambda D_V \mathbf{u} \tag{7.10}$$

Thus, the optimal U and V can be obtained by iteratively computing the generalized eigenvectors of (7.9) and (7.10). In our experiments, U is initially set to the identity matrix. It is easy to show that the matrices $D_U, D_V, D_U - S_U$, and $D_V - S_V$ are all symmetric and positive semi-definite.

7.3 Face Recognition with Tensor Representation

In this section, several experiments are carried out to show the efficiency and effectiveness of our proposed algorithm for face recognition. We compare our algorithm with the Eigenface (PCA) [72], Fisherface (LDA) [9], and Laplacianface (LPP) [34] methods, three of the most popular linear methods for face recognition.

Two face databases were used. The first one is the PIE (Pose, Illumination, and Experience) database from CMU, and the second one is the ORL database. In all the experiments, preprocessing to locate the faces was applied. Original images were normalized (in scale and orientation) such that the two eyes were aligned at the same position. Then, the facial areas were cropped into the final images for matching. The size of each cropped image in all the experiments is 32×32 pixels, with 256 gray levels per pixel. No further preprocessing is done. For the Eigenface, Fisherface, and Laplacianface methods, the image is represented as a 1024-dimensional vector, while in our algorithm the image is represented as a (32×32)-dimensional matrix, or the second order tensor. The nearest neighbor classifier is used for classification for its simplicity.

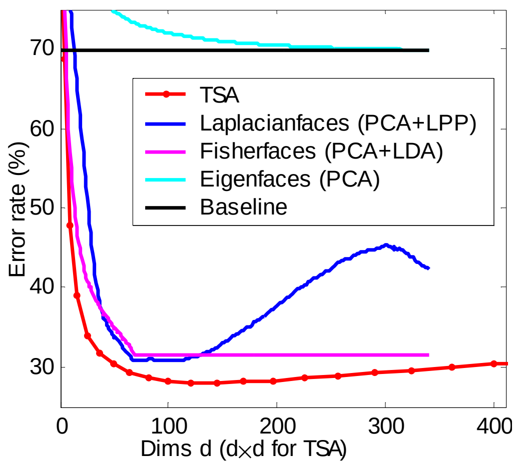
In short, the recognition process has three steps. First, we calculate the face subspace from the training set of face images; then the new face image to be identified is projected into d-dimensional subspace (PCA, LDA, and LPP) or $(d \times d)$-dimensional tensor subspace (TSA); finally, the new face image is identified by nearest neighbor classifier. In our TSA algorithm, the number of iterations is taken to be 3.

7.3.1 Experiments on PIE Database

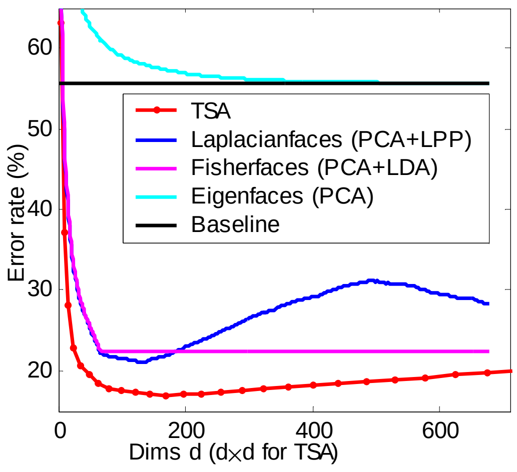
The CMU PIE face database contains 68 subjects with 41,368 face images as a whole. The face images were captured by 13 synchronized cameras and 21 flashes, under varying pose, illumination and expression. We choose the five near frontal poses (C05, C07, C09, C27, C29) and use all the images under different illuminations and expressions, thus we get 170 images for each individual. For each individual, $l(= 5, 10, 20, 30)$ images are randomly selected for training and the rest are used for testing.

The training set is utilized to learn the subspace representation of the face manifold by using Eigenface, Fisherface, Laplacianface and our algorithm. The testing images are projected into the face subspace in which recognition is then performed. For each given l, we average the results over 20 random splits. It would be important to note that the Laplacianface algorithm and our algorithm share the same graph structure as defined in Eqn.(7.3).

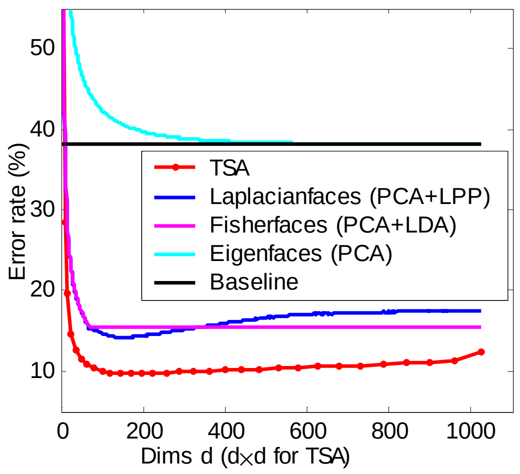
Figure 7.1 shows the plots of error rate versus dimensionality reduction for the Eigenface, Fisherface, Laplacianface, TSA and baseline methods. For the baseline method, the recognition is simply performed in the original 1024-dimensional image space without any dimensionality reduction. Note that, the upper bound of the dimensionality of Fisherface is $c-1$ where c is the number of individuals. For our TSA algorithm, we only show its performance in the $(d \times d)$-dimensional tensor subspace, say, 1, 4, 9, etc. As can be seen, the performance of the Eigenface, Fisherface, Laplacianface, and TSA algorithms varies with the number of dimensions. We show the best results obtained by them in Table 7.1 and the corresponding face subspaces



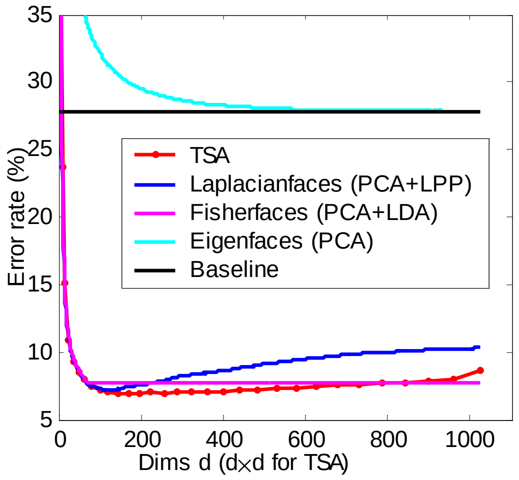
(a) 5 Train



(b) 10 Train



(c) 20 Train



(d) 30 Train

Figure 7.1: Error rate vs. dimensionality reduction on PIE database

are called optimal face subspace for each method.

It is found that our method outperforms the other four methods with different numbers of training samples (5, 10, 20, 30) per individual. The Eigenface method performs the worst. It does not obtain any improvement over the baseline method. The Fisherface and Laplacianface methods perform comparatively to each each. The dimensions of the optimal subspaces are also given in Table 7.1.

As we have discussed, TSA can be implemented very efficiently. We show the running time in seconds for each method in Table 7.1. As can be seen, TSA is much faster than the Eigenface, Fisherface and Laplacianface methods. All the algorithms

Table 7.1: Performance comparison on PIE database

Method	5 Train			10 Train		
	error	dim	time(s)	error	dim	time(s)
Baseline	69.9%	1024	-	55.7%	1024	-
Eigenfaces	69.9%	338	0.907	55.7%	654	5.297
Fisherfaces	31.5%	67	1.843	22.4%	67	9.609
Laplacianfaces	30.8%	67	2.375	21.1%	134	11.516
TSA	**27.9%**	**11^2**	**0.594**	**16.9%**	**13^2**	**2.063**

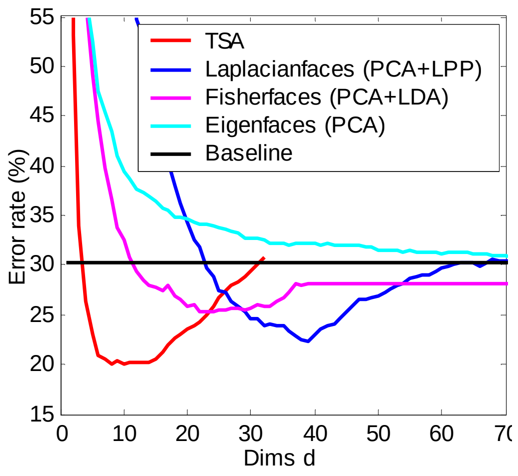
Method	20 Train			30 Train		
	error	dim	time(s)	error	dim	time(s)
Baseline	38.2%	1024	-	27.9%	1024	-
Eigenfaces	38.1%	889	14.328	27.9%	990	15.453
Fisherfaces	15.4%	67	35.828	7.77%	67	38.406
Laplacianfaces	14.1%	146	39.172	7.13%	131	47.610
TSA	**9.64%**	**13^2**	**7.125**	**6.88%**	**12^2**	**15.688**

were implemented in Matlab 6.5 and run on a Intel P4 2.566GHz PC with 1GB memory.

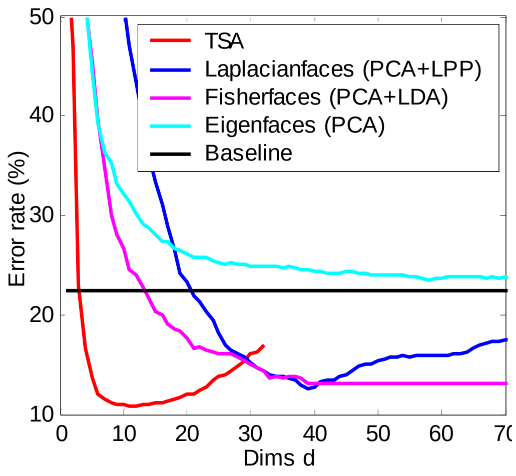
7.3.2 Experiments on ORL Database

The ORL (Olivetti Research Laboratory) face database is used in this test. It consists of a total of 400 face images, of a total of 40 people (10 samples per person). The images were captured at different times and have different variations including expressions (open or closed eyes, smiling or non-smiling) and facial details (glasses or no glasses). The images were taken with a tolerance for some tilting and rotation of the face up to 20 degrees. For each individual, $l(= 2, 3, 4, 5)$ images are randomly selected for training and the rest are used for testing.

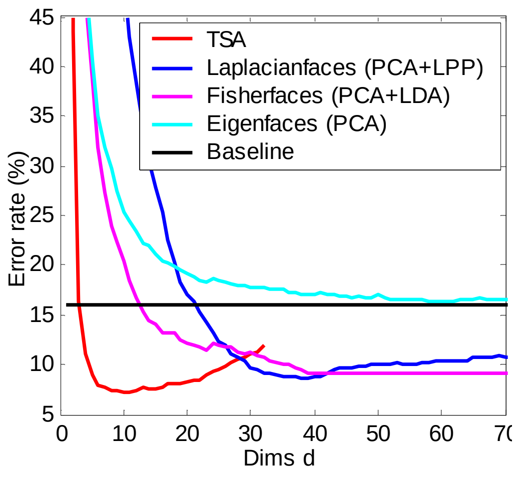
The experimental design is the same as that in the last subsection. For each given l, we average the results over 20 random splits. Figure 7.3.2 shows the plots of error rate versus dimensionality reduction for the Eigenface, Fisherface, Laplacianface, TSA and baseline methods. Note that, the presentation of the performance of the TSA algorithm is different from that in the last subsection. Here, for a given d, we show its performance in the $(d \times d)$-dimensional tensor subspace. The reason is for better



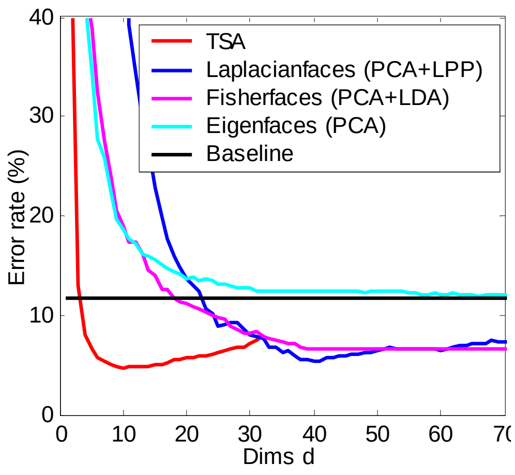
(a) 2 Train



(b) 3 Train



(c) 4 Train



(d) 5 Train

Figure 7.2: Error rate vs. dimensionality reduction on ORL database

Table 7.2: Performance comparison on ORL database

Method	2 Train			3 Train		
	error	dim	time	error	dim	time
Baseline	30.2%	1024	-	22.4%	1024	-
Eigenfaces	30.2%	79	38.13	22.3%	113	85.16
Fisherfaces	25.2%	23	60.32	13.1%	39	119.69
Laplacianfaces	22.2%	39	62.65	12.5%	39	136.25
TSA	**20.0%**	**10^2**	**65.00**	**10.7%**	**11^2**	**135.93**

	4 Train			5 Train		
Method	error	dim	time	error	dim	time
Baseline	16.0%	1024	-	11.7%	1024	-
Eigenfaces	15.9%	122	141.72	11.6%	182	224.69
Fisherfaces	9.17%	39	212.82	6.55%	39	355.63
Laplacianfaces	8.54%	39	248.90	5.45%	40	410.78
TSA	**7.12%**	**10^2**	**201.40**	**4.75%**	**10^2**	**302.97**

comparison, since the Eigenface and Laplacianface methods start to converge after 70 dimensions and there is no need to show their performance after that. The best result obtained in the optimal subspace and the running time (millisecond) of computing the eigenvectors for each method are shown in Table 7.2.

As can be seen, our TSA algorithm performed the best in all the cases. The Fisherface and Laplacianface methods performed comparatively to our method, while the Eigenface method performed poorly.

7.4 Image Clustering with Tensor Representation

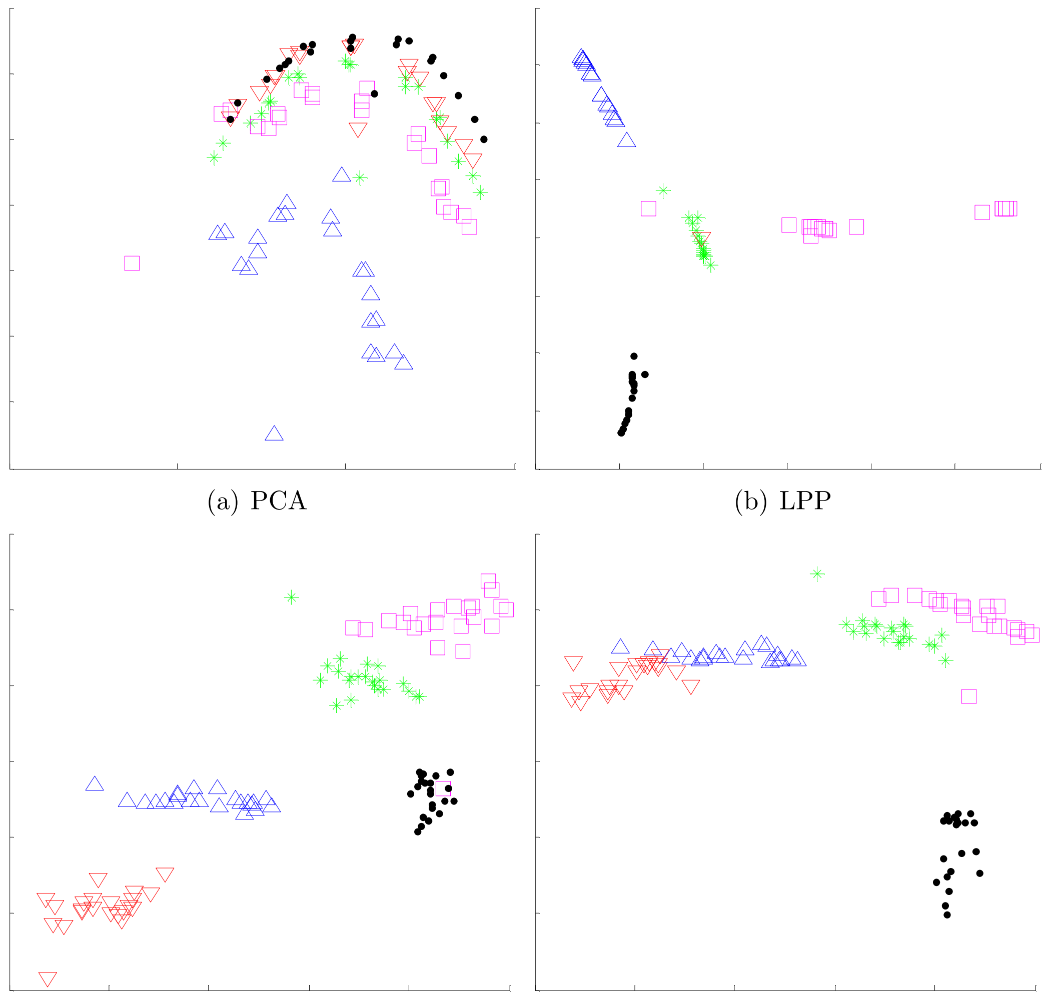
Once we project the images into a tensor subspace, clustering can be performed in such a lower dimensional space. Some typical clustering algorithms include k-means, Gaussian Mixture Model, and spectral clustering [56], [68]. Spectral clustering have attracted considerable attention in recent years. It can be thought of as a combination of spectral dimensionality reduction [10] and traditional clustering algorithms like k-means. In this work, we apply k-means in the tensor subspace for clustering. The TSA approach may be interpreted as a search for good spectral partitioning functions by restricting our search to *multilinear* functions alone. The objective function of TSA is actually a min-cut like graph partitioning criteria. In this sense, our method can be thought of as a multilinear spectral clustering. The relationship between spectral clustering and dimensionality reduction can be found in [10].

7.4.1 Data Preparation

The database used in our experiment is the PIE (Pose, Illumination, and Experience) database from CMU. It contains 68 subjects with 41,368 face images as a whole. The face images were captured under varying pose, illumination and expression. We fixed the pose and expression. Thus, for each subject, we got 22 images under different lighting conditions. Figure 7.4 shows some sample images for a certain subject. Preprocessing to locate the faces was applied. Original images were normalized (in scale and orientation) such that the two eyes were aligned at the same position. Then, the facial areas were cropped into the final images for matching. The size of each cropped image in all the experiments is 32×32 pixels, with 256 gray levels per pixel. No further preprocessing is done. For traditional learning algorithms (PCA and LPP), the image is represented as a 1024-dimensional vector. For TSA, the image is represented as a (32×32)-dimensional matrix, or the second order tensor.

7.4.2 2-D Visualization of Image Set

As we described previously, PCA, LPP and TSA are different dimensionality reduction algorithms. In this subsection, we use them to project the images into a 2-dimensional subspace for visualization. We randomly selected 5 classes for this test. Figure 7.3 shows the 2D embedding results. In this example, we did not show the embedding result of Laplacian Eigenmaps [10] because it gives the same result as LPP, please see Proposition 2 for details. For TSA method, we can project the images into either $\mathcal{R}^1 \otimes \mathcal{R}^2$ or $\mathcal{R}^2 \otimes \mathcal{R}^1$, both of which are 2-dimensional spaces. $\mathcal{R}^1 \otimes \mathcal{R}^2$ is formed by the projection $\mathbf{u}_1^T X[\mathbf{v}_1, \mathbf{v}_2]$ and $\mathcal{R}^2 \otimes \mathcal{R}^1$ is formed by the projection $[\mathbf{u}_1, \mathbf{u}_2]^T X \mathbf{v}_1$. As can be seen, PCA performs the worst. It fails to distinguish the different classes, and the five classes are mixed together. TSA performed marginally better than LPP. For LPP, we can see that there are two classes mixed together.



(a) PCA (b) LPP

(c) Tensor subspace $\mathcal{R}^1 \otimes \mathcal{R}^2$ (d) Tensor subspace $\mathcal{R}^2 \otimes \mathcal{R}^1$

Figure 7.3: 2D visualization of image set

Figure 7.4: Sample face images from CMU PIE database. For each subject, there are 22 face images under different lighting conditions and fixed pose (C27) and expression.

Table 7.3: Clustering performance comparisons on CMU PIE database

k	Accuracy (%)				
	Kmeans	PCA	LPP	NCut	TSA
5	49.3	51.3	96.6	96.6	**99.95**
10	39.7	40.8	86.1	86.1	**92.95**
30	34.9	35.4	77.8	77.8	**84.32**
68	33.6	34.4	74.5	73.5	**82.23**

k	Mutual Information (%)				
	Kmeans	PCA	LPP	NCut	TSA
5	46.7	47.8	97.0	97.0	**99.88**
10	50.1	51.1	92.9	92.9	**96.95**
30	56.1	56.9	90.9	90.9	**94.95**
68	62.6	63.9	91.3	90.6	**95.20**

Clearly, these two classes will be grouped together when clustering is performed. This illustrative example shows that TSA can have more discriminating power than PCA and LPP.

7.4.3 Clustering Results

We compared the following five algorithms for image clustering:

- baseline: k-means in the original space (K-means)
- our algorithm: TSA+k-means (TSA)
- PCA+k-means (PCA)
- LPP+k-means (LPP, [88])

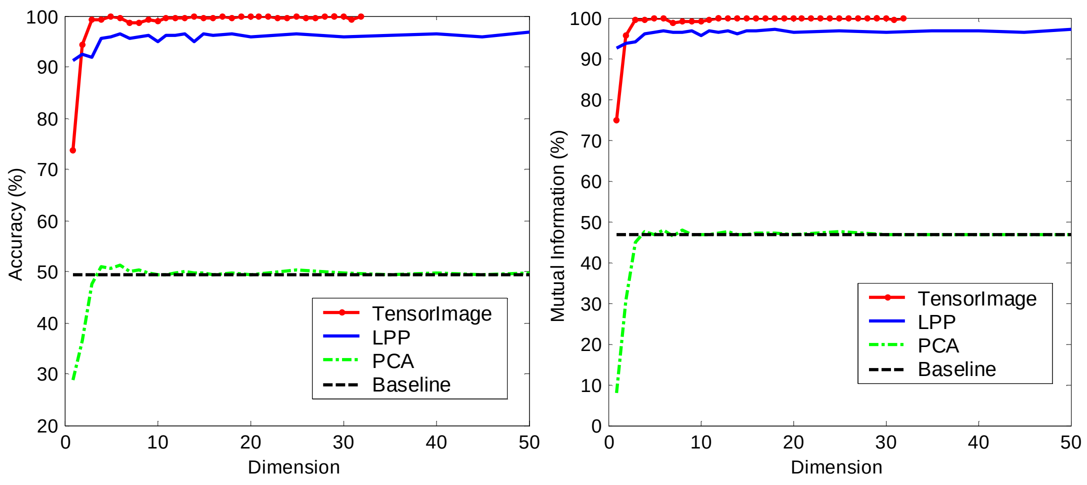


Figure 7.5: Clustering results on 5 randomly chosen classes

- Normalized Cut (NCut, [68])

Note that, TSA, PCA, and LPP are all linear algorithms. Normalized cut (NCut, [68]) is nonlinear. NCut can be thought of as a combination of Laplacian Eigenmaps and k-means. We tested these algorithms on several cases. For each case, $k(= 5, 10, 30, 68)$ classes were randomly selected from the data corpus. The data points and the cluster number k are provided to the clustering algorithms. The clustering result is evaluated by comparing the obtained label of each data point with that provided by the data corpus. The evaluation metrics used here are the same as those described in Chapter 6.

The evaluations were conducted with different numbers of clusters. For each given class number k, k classes were randomly selected from the database. This process were repeated 50 times, and the average performance was computed. For each single test (given k classes of face images), we applied the above five methods. For each method, the k-means step was repeated 10 times with different initializations and the best result was recorded. For LPP, PCA, NCut, and TSA, they all need to estimate the dimensionality of the subspace. In general, their performance varies with the dimensionality of the subspace. For LPP, PCA and NCut, the images were projected into $\mathbb{R}^d(d < 1024)$. For TSA, the images were projected into $\mathcal{R}^d \otimes \mathcal{R}^d(1 \leq d < 32)$. Notice that, for TSA, the images can be actually projected into $\mathcal{R}^{d_1} \otimes \mathcal{R}^{d_2}(1 \leq d_1, d_2 <$

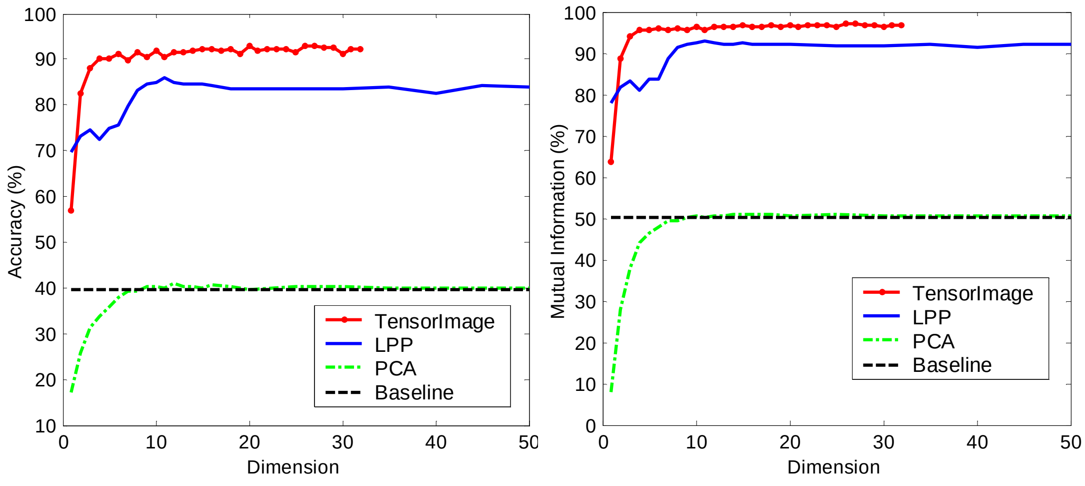


Figure 7.6: Clustering results on 10 randomly chosen classes

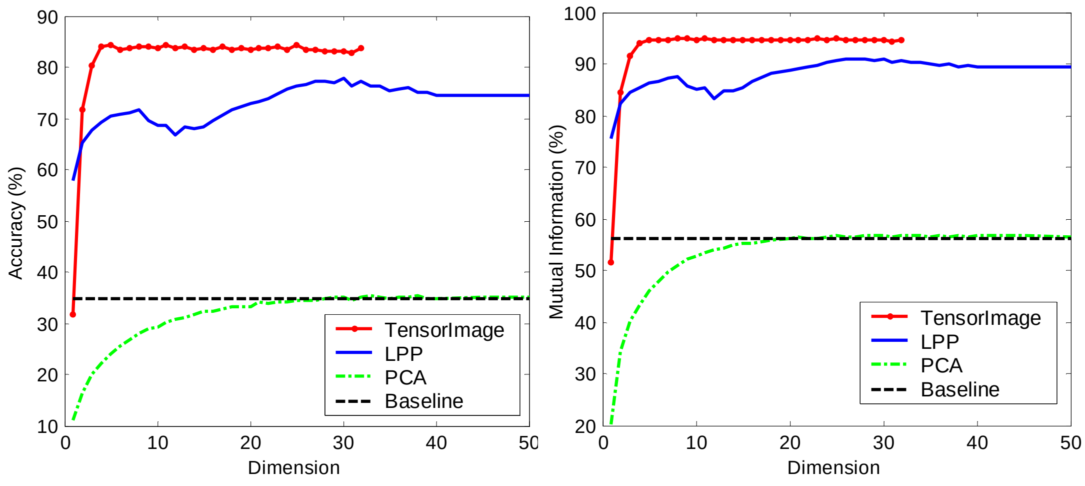


Figure 7.7: Clustering results on 30 randomly chosen classes

32). In our experiment, we set $d_1 = d_2$ for the sake of simplicity. Figure 7.5-7.8 show the clustering performance of these algorithms as a function of the dimensionality of the subspace (d). Table 1 shows the best performance obtained by each algorithm. As can be seen, our clustering algorithm consistently outperformed PCA and LPP based clustering algorithms. Note that, when the number of classes (k) is 5, 10, or 30, the total number of images ($22 \times k$) is less than the number of features (1024). Therefore, NCut has the same result as LPP as suggested by Proposition 2. Moreover, the performance of PCA based clustering algorithm is almost the same as that of baseline. This shows that PCA fails to discover the intrinsic class structure of the

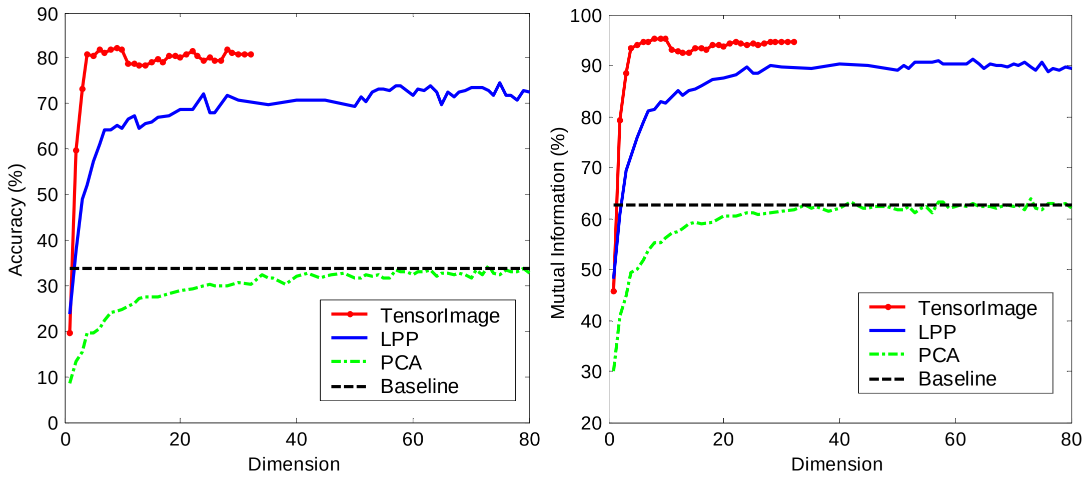


Figure 7.8: Clustering results on 68 randomly chosen classes

image database. Finally, it can be seen that LPP, NCut and TSA achieved their best results at very low dimensions. This indicates that dimensionality reduction is a necessary preprocessing step for data clustering and classification.

CHAPTER 8
CONCLUSIONS

In this thesis, we propose a new linear dimensionality reduction algorithm called Locality Preserving Projections. It is based on the same variational principle that gives rise to the Laplacian Eigenmap [10]. As a result it has similar locality preserving properties.

Our approach also has several possible advantages over recent nonparametric techniques for global non-linear dimensionality reduction such as [10][61][71]. It yields a map which is simple, linear, and defined everywhere (and therefore on novel test data points). The algorithm can be easily kernelized yielding a natural non-linear extension.

Performance improvement of this method over Principal Component Analysis is demonstrated through several experiments. Though our method is a linear algorithm, it is adapted as far as possible to discover the nonlinear structure of the data manifold. LPP can be performed in either supervised or unsupervised manner. When the label information is available, it can be easily incorporated into the nearest-neighbor graph.

In this thesis, we have shown that LPP is potentially applicable to face analysis and document indexing. We have also applied LPP to handwritten digit recognition on MNIST database. However, on this database, LDA outperforms LPP. One reason might be that, on this database the nearest neighbor graph constructed can not accurately reflect the manifold structure of the data space.

One of the major bottlenecks of LPP is nearest neighbor search. Though there have been a lot of work on this topic, most of them have a assumption that the dimensionality of the data space is typically less than 20. However, in many real world applications such as face recognition and information retrieval, the dimensionality ranges from hundreds to thousands. The high dimensional nearest neighbor search problem will be investigated in our future work.

REFERENCES

[1] Yale univ. face database. http://cvc.yale.edu/projects/yalefaces/yalefaces.html, 2002.

[2] A. L. A. Shashua and S. Avidan. Manifold pursuit: A new approach to appearance based recognition. In *International Conference on Pattern Recognition*, Quebec, Canada, Auguet 2002.

[3] R. Ando. Latent semantic space: Iterative scaling improves precision of inter-document similarity measurement. In *Proceedings of the ACM SIGIR Conference on Information Retrieval*, 2000.

[4] R. Ando and L. Lee. Iterative residual rescaling: An analysis and generalization. In *Proceedings of the ACM SIGIR Conference on Information Retrieval*, 2001.

[5] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press / Addison-Wesley, 1999.

[6] B. T. Bartell, G. W. Cottrell, and R. K. Belew. Latent semantic indexing is an optimal special case of multidimensional scaling. In *Proceedings of the ACM SIGIR Conference on Information Retrieval*, Copenhagen, Denmark, 1992.

[7] A. U. Batur and M. H. Hayes. Linear subspace for illumination robust face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[8] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10):2385–2404, 2000.

[9] P. Belhumeur, J. Hepanha, and D. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *IEEE Trans. on PAMI*, 19(7):711–720, 1997.

[10] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, 2001.

[11] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *Advances in Neural Information Processing Systems 16*, 2003.

[12] J. L. Bentley. Multidimensional divide and conquer. *Communications of the ACM*, 23(4):214–229, 1990.

[13] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proc. Of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 245–250, 2003.

[14] C. Blake and C. Merz. Uci repository of machine learning databases. http://www.ics.uci.edu/ mlearn/MLRepository.html, 1998.

[15] M. Brand. Charting a manifold. In *Advances in Neural Information Processing Systems 16*, 2003.

[16] Y. Chang, C. Hu, and M. Turk. Manifold of facial expression. In *Proc. IEEE International Workshop on Analysis and Modeling of Faces and Gestures*, Nice, France, October 2003.

[17] F. R. K. Chung. *Spectral Graph Theory*, volume 92 of *Regional Conference Series in Mathematics*. 1997.

[18] D. Cohn. Informed projections. In *Advances in Neural Information Processing Systems 15*, 2002.

[19] S. Dasgupta. Experiments with random projection. In *Sixteenth Conference on Uncertainty in Artificial Intelligence*, 2000.

[20] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[21] C. H. Ding. A similarity based probability model for latent semantic indexing. In *Proceedings of the ACM SIGIR Conference on Information Retrieval*, 1999.

[22] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley-Interscience, Hoboken, NJ, 2nd edition, 2000.

[23] S. T. Dumais and J. Nielsen. Automating the assignment of submitted manuscripts to reviewers. In *Proceedings of the ACM SIGIR Conference on Information Retrieval*, Copenhagen, Denmark, 1992.

[24] J. G. Dy and C. E. Brodley. Feature subset selection and order identification for unsupervised leanring. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

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

[26] A. Fitzgibbon and A. Zisserman. Joint manifold distance: a new approach to appearance based clustering. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2003.

[27] P. W. Foltz and S. T. Dumais. Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM*, 35(12):51–60, 1992.

[28] Y. Grandvalet and S. Canu. Adaptive scaling for feature selection in svms. In *Advances in Neural Information Processing Systems 15*, 2002.

[29] R. Gross, J. Shi, and J. Cohn. Where to go with face recognition. In *Third Workshop on Empirical Evaluation Methods in Computer Vision*, Kauai, Hawaii, December 2001.

[30] S. Guattery and G. L. Miller. Graph embeddings and laplacian eigenvalues. *SIAM Journal on Matrix Analysis and Applications*, 21(3):703–723, 2000.

[31] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* New York: Springer-Verlag, 2001.

[32] S. Haykin. *Neural Networks, A Comprehensive Fundation.* Prentice Hall, 2nd edition, 1998.

[33] X. He and P. Niyogi. Locality preserving projections. *Advances in Neural Information Processing Systems*, 16, 2003.

[34] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang. Face recognition using laplacianfaces. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(3):328–340, 2005.

[35] J. S. U. Hjorth. *Computer Intensive Statistical Methods Validation, Model Selection, and Bootstrap.* London: Chapman and Hall, 1994.

[36] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman. Clustering appearances of objects under varying illumination conditions. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2003.

[37] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the ACM SIGIR Conference on Information Retrieval*, Berkeley, California, 1999.

[38] C. L. Isbell and P. Viola. Restructuring sparse high dimensional data for effective retrieval. In *Advances in Neural Information Processing Systems*, 1999.

[39] Q. L. J. Ye, R. Janardan. Two-dimensional linear discriminant analysis. *Advances in Neural Information Processing Systems*, 17, 2004.

[40] I. T. Jolliffe. *Principal Component Analysis.* Springer-Verlag, New York, 1989.

[41] J. M. Kleinberg. Two algorithms for nearest neighbor search in high dimensions. In *Proc. 29th ACM Symposium on Theory of Computing (SOTC)*, 1997.

[42] R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[43] T. G. Kolda and D. P. O'Leary. A semi-discrete matrix decomposition for latent semantic indexing in information retrieval. *ACM Transactions on Information Systems*, 16(4):322–346, 1998.

[44] K. Lang. Learning to filter netnews. In *Proc. of the 12th Int. Conf. on Machine Learning*, 1995.

[45] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[46] J. M. Lee. *Introduction to Smooth Manifolds.* Springer-Verlag New York, 2002.

[47] K.-C. Lee, J. Ho, M.-H. Yang, and D. Kriegman. Video-based face recognition using probabilistic appearance manifolds. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2003.

[48] A. Levin and A. Shashua. Principal component analysis over continuous subspaces and intersection of half-spaces. In *Proc. of the European Conference on Computer Vision*, 2002.

[49] J. Lim, J. Ho, M.-H. Yang, K.-C. Lee, and D. Kriegman. Image clustering with metric, local linear structure and affinity symmetry. In *The 8th European Conference on Computer Vision*, 2004.

[50] Q. Liu, R. Huang, H. Lu, and S. Ma. Face recognition using kernel based fisher discriminant analysis. In *Proc. of the fifth International Conference on Automatic Face and Gesture Recognition*, Washington, D. C., May 2002.

[51] L. Lovasz and M. Plummer. *Matching Theory.* Akadémiai Kiadó, North Holland, Budapest, 1986.

[52] A. M. Martinez and A. C. Kak. Pca versus lda. *IEEE Trans. on PAMI*, 23(2):228–233, 2001.

[53] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *IEEE Trans. on PAMI*, 19(7):696–710, 1997.

[54] B. Mohar. Some applications of laplace eigenvalues of graphs. In *In G. Hahn and G. Sabidussi, editors, Graph Symmetry: Algebraic Methods and Applications*, 1997.

[55] H. Murase and S. K. Nayar. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14, 1995.

[56] A. Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, 2001.

[57] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: a probabilistic analysis. In *Proc. 17th ACM Symp. Principles of Database Systems*, Seattle, 1998.

[58] P. J. Phillips. Support vector machines applied to face recognition. *Advances in Neural Information Processing Systems*, 11:803–809, 1998.

[59] M. Plutowski, S. Sakata, and H. White. Cross-validation estimates imse. *Neural Information Processing Systems*, 6, 1994.

[60] V. Roth and T. Lange. Feature selection in clustering problems. In *Advances in Neural Information Processing Systems 16*, 2003.

[61] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[62] L. S. Sam Roweis and G. Hinton. Global coordination of local linear models. In *Advances in Neural Information Processing Systems 14*, 2001.

[63] L. K. Saul and S. T. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

[64] B. Scholkopf, A. Smola, and K. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[65] H. S. Seung and D. D. Lee. The manifold ways of perception. *Science*, 290(12), 2000.

[66] T. Shakunaga and K. Shigenari. Decomposed eigenface for face recognition under various lighting conditions. In *IEEE Conference on Computer Vision and Pattern Recognition*, Hawaii, December 2001.

[67] J. Shao and D. Tu. *The Jackknife and Bootstrap.* New York: Springer-Verlag, 1995.

[68] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on PAMI*, 22(8):888–905, 2000.

[69] T. Sim, S. Baker, and M. Bsat. The cmu pose, illuminlation, and expression database. *IEEE Trans. on PAMI*, 25(12):1615–1618, 2003.

[70] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization. *Journal of the Optical Society of America*, 4:519–524, 1987.

[71] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(12):2319–2323, 2000.

[72] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[73] M. Turk and A. P. Pentland. Face recognition using eigenfaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, Maui, Hawaii, 1991.

[74] P. Vaidya. An $o(n \log n)$ algorithm for the all-nearest-neighbor problem. *Discrete and Computtational Geometry*, 4:101–115, 1989.

[75] N. Vasconcelos. Feature selection by maximum marginal diversity. In *Advances in Neural Information Processing Systems 15*, 2002.

[76] M. A. O. Vasilescu and D. Terzopoulos. Multilinear subspace analysis for image ensembles. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[77] Y. Weiss. Segmentation using eigenvectors: a unifying view. In *IEEE International Conference on Computer Vision*, 1999.

[78] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. In *Advances in Neural Information Processing Systems 14*, 2001.

[79] L. Wiskott, J. Fellous, N. Kruger, and C. Malsburg. Face recognition by elastic bunch graph matching. *IEEE Trans. on PAMI*, 19:775–779, 1997.

[80] R. Xiao, L. Zhu, and H.-J. Zhang. Boosting chain learning for object detection. In *IEEE International Conference on Computer Vision*, Nice, France, 2003.

[81] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of ACM SIGIR*, 2003.

[82] S. Yan, M. Li, H.-J. Zhang, and Q. Cheng. Ranking prior likelihood distributions for bayesian shape localization framework. In *IEEE International Conference on Computer Vision*, Nice, France, 2003.

[83] J. Yang, Y. Yu, and W. Kunz. An efficient lda algorithm for face recognition. In *The sixth International Conference on Control, Automation, Robotics and Vision*, Singapore, 2000.

[84] J. Yang, D. Zhang, A. Frangi, and J. Yang. Two-dimensional pca: a new approach to appearance-based face representation and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1), January 2004.

[85] M.-H. Yang. Kernel eigenfaces vs. kernel fisherfaces: Face recognition using kernel methods. In *Proc. of the fifth International Conference on Automatic Face and Gesture Recognition*, Washington, D. C., May 2002.

[86] H. Zha and Z. Zhang. Isometric embedding and continuum isomap. In *Proc. of the twentieth Internation Conference on Machine Learning*, 2003.

[87] W. Zhao, R. Chellappa, and P. J. Phillips. Subspace linear discriminant analysis for face recognition. Technical Report CAR-TR-914, Center for Automation Research, University of Maryland, 1999.

[88] X. Zheng, D. Cai, X. He, W.-Y. Ma, and X. Lin. Locality preserving clustering for image database. In *Proceedings of the ACM Conference on Multimedia*, October 2004.