A NOVEL SUPERVISED DIMENSIONALITY REDUCTION METHOD: INTEGRATING PCA WITH SVM

by

FAEZEH SOLEIMANI

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON
August 2021

A NOVEL SUPERVISED DIMENSIONALITY REDUCTION METHOD:

INTEGRATING PCA WITH SVM

The members of the Committee approve the doctoral
dissertation of FAEZEH SOLEIMANI

Dr. Li Wang _____

(Supervising Professor)

Dr. Ren-Cang Li _____

(Supervising Professor)

Dr. Tuncay Aktosun _____

Dr. Hristo V. Kojouharov _____

Dean of the Graduate School _____

To my parents, Samaneh and Masoud.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisors Dr. Li Wang and Dr. Ren-Cang Li. This journey of my research career at the University of Texas at Arlington (UTA) would not have been possible without their continuous support, patience, and encouragement. I can't imagine coming to this point in my research without the benefit of their immense knowledge and intellectual curiosity. They have helped me grow personally, academically, research-wise, and professionally.

I am very thankful to my dissertation committee members Dr. Tuncay Aktosun and Dr. Hristo V. Kojouharov for their encouragement and positive feedback. My professors and advisors, specially Dr. Barbara Shipman, were always there to support me with their insightful comments, recommendations, and constructive feedback. Their guidance and mentorship have benefited me in every step. I am forever grateful.

I also wish to thank everyone in the Mathematics department at UTA, for being supportive, encouraging and kind since day one. I could not wish a better environment than here for achieving a doctorate degree. Kiran Mainali, Saul Covarrubias, and Talon Johnson, my academic siblings, deserve special thanks for being great friends and coworkers. We have broadened our knowledge with insightful research discussions and have helped each other in every way we could during our studies. I am also grateful to my friends who have been there to help me in any way possible until the last moment.

Last but not least, I would like to express my sincere gratitude and indebtedness to my parents, my husband, and my sister for their love, support, and encouragement throughout the years. Without their presence, none of my success would have been possible.

July 1st, 2021

ABSTRACT

A NOVEL SUPERVISED DIMENSIONALITY REDUCTION METHOD:
INTEGRATING PCA WITH SVM

FAEZEH SOLEIMANI, Ph.D.

The University of Texas at Arlington, 2021

Supervising Professors: Dr. Li Wang & Dr. Ren-Cang Li

Data curation and storage methods have changed over the past few decades with the use of new technologies, and gathering data on a huge number of features (dimensions) is now very common among diverse scientific and engineering fields. Prior to classification or regression, dimensionality reduction is necessary to eliminate irrelevant features and to deal with data with high dimensions. A number of numerical methods have already been proposed to reduce the dimension of data, for example, Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Supervised Principal Component Analysis (SPCA).

In this dissertation, we will introduce a novel way of reducing dimensionality and classifying data simultaneously through a supervised approach that reduces dimension while classifying data. The objective of our model is to determine the projection matrix utilized in the dimensionality reduction procedure, as well as to determine the hyperplane of the classifier used for data classification. Since the supervised model is learning both the representation of the low-dimensional data and the classification simultaneously, our supervised model has the advantage of high accuracy as well as

effective representation. Additionally, our model is capable of performing multi-class classification, i.e., it is capable of classifying data with more than two categories. Due to our model's ability to use nonlinear mappings, we can also apply it to data sets with nonlinear and complex structures. Simulating the model and comparing it with the state-of-the-art dimensionality reduction and classification techniques demonstrate the model's effectiveness and efficiency.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ALGORITHMS

LIST OF TABLES

LIST OF ABBREVIATIONS

DR	Dimensionality Reduction
SVM	Support Vector Machine
PCA	Principal Component Analysis
KPCA	Kernel Principal Component Analysis
LDA	Linear Discriminant Analysis
SPCA	Supervised Principal Component Analysis
KSPCA	Kernel Supervised Principal Component Analysis
RBF	Radial Basis Function
ANN	Artificial Neural Network
HSIC	Hilbert-Schmidt's Independence Criterion
KKT	Karush-Kuhn-Tucker
P.S.D	Positive Semi-definite
P.D	Positive Definite
SVD	Singular Value Decomposition

RE	Reconstruction Error
LIBSVM	Library for Support Vector Machine
kNN	k-Nearest Neighbor
SMO	Sequential Minimal Optimization

LIST OF SYMBOLS

X	Data matrix
P	Projection matrix
K	Kernel matrix
I	Identity matrix
Σ	Diagonal matrix
$\mathcal{H}$	Hilbert space
min	Minimize
max	Maximize
$\mathrm{Tr}(A)$	Trace of matrix A
A^{-1}	Inverse of matrix A
A^T	Transpose of matrix A
$\|x\|_2$	Euclidean norm of vector x
$\|X\|_F$	Frobenius norm of matrix X
e	Vector of all ones
$\nabla(f)$	Gradient of the function f
$\mathrm{Cov}(X)$	Covariance matrix of X
x_i	The ith data point

z_i	Label of the ith data point
ϵ	Misclassification degree
C	SVM regularization parameter
w	Normal vector
x^k	Value of x in the kth iteration step
$\mathrm{sgn}(x)$	The sign function acting on x
$DF(x)$	The decision function acting on x
Φ	The nonlinear mapping
μ	Total mean of the data
$\lambda(A)$	Eigenvalue of matrix A
κ	The kernel function
$d(\cdot,\cdot)$	Distance
$\langle\cdot,\cdot\rangle$	Inner product
Tol	Tolerance
$\mathcal{O}$	Set of orthonormal matrices

CHAPTER 1

INTRODUCTION

1.1 Motivations

Real-world data with high dimensions such as those from fMRI scans and speech signals are generated voluminously from various sources. Capabilities to store large data sets and advances in data collection have led to an information overload in most scientific disciplines. On a daily basis, more and more observations and simulations are conducted by researchers working in domains as diverse as biology, mathematics, computer science, engineering, astronomy, and economics. Such data typically possesses a large number of features/variables and poses various mathematical challenges. The number of features/variables that are measured on each observation defines the "*dimension*" of the data. One of the problems with high-dimensional data is that, in many cases, although the dimension of the data is high, only a few key features are of critical importance for understanding the underlying phenomena of interest and modeling tasks. Another challenge is that handling such huge data is a burdensome task and the learning task becomes significantly more difficult with an increase in the number of features. Due to the increase in the number of variables associated with each observation, most of the traditional statistical methods break down and fail to perform well. "Curse of Dimensionality" [1] is a serious problem caused by the increase in the volume associated with adding extra dimensions to space; see Figure 1.1.

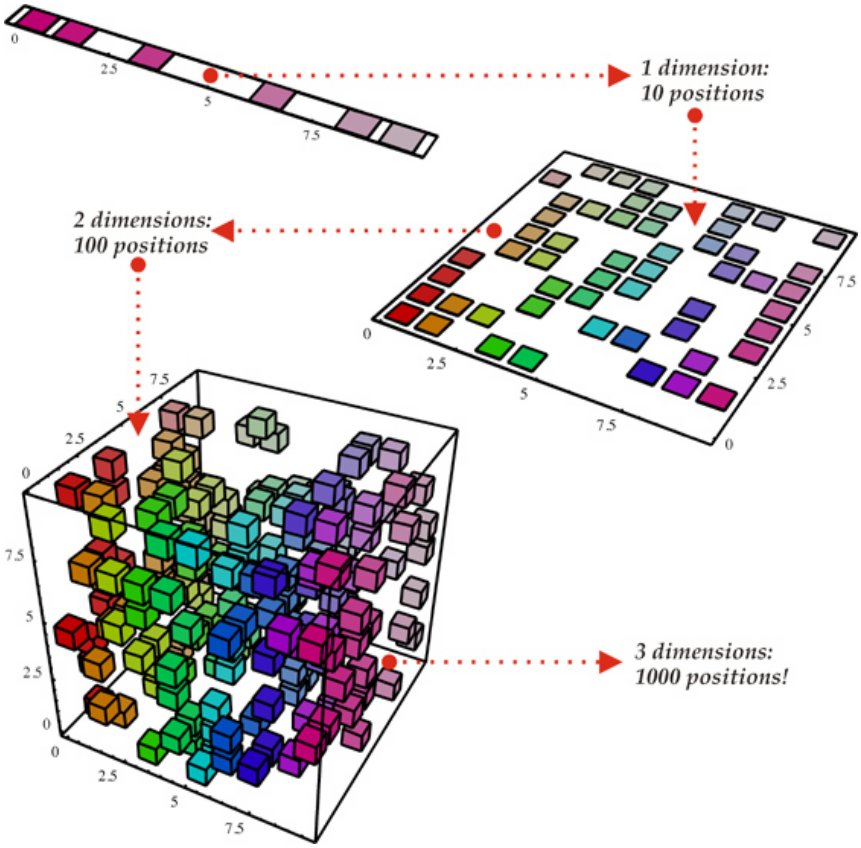


Figure 1.1: The increase in the volume associated with adding extra dimensions to space caused by the Curve of Dimensionality [3].

1.2 Dimensionality reduction problem

In order to handle high-dimensional data, to remove irrelevant features, and to mitigate the curse of dimensionality and other undesired properties of high-dimensional spaces [2], the dimension of the original data in the high-dimensional space should be reduced prior to any modeling of the data, i.e., a subset of features needs to be carefully chosen and used. Transforming high-dimensional data into a representation of reduced dimensionality is called "Dimensionality Reduction (DR)" which will help us to identify the key features of the data; see Figure 1.2.

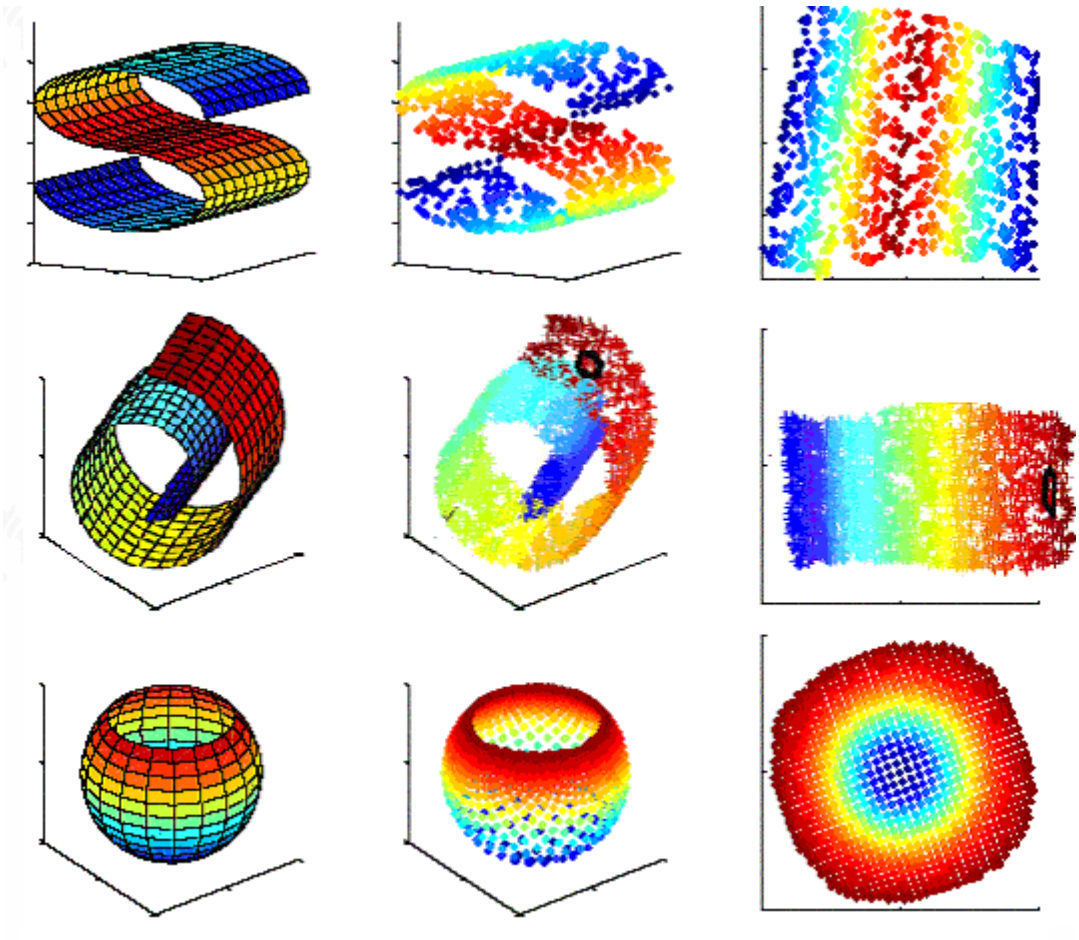


Figure 1.2: Mapping some 3-dimensional data into their representation of reduced dimensionality $d = 2$ [4].

The goal of dimensionality reduction techniques is to map high-dimensional data to a low-dimensional space such that certain properties of the original data are preserved. Neighborhood information [5, 6], local tangent space [7], global geometry, and distances between data points [8, 9] are among certain properties that dimensionality reduction methods aim to preserve. There are different ways to reduce the dimension of the data. However, the most popular techniques in DR are feature selection and feature extraction. Feature selection techniques try to select a subset of the current features [10, 11], while feature extraction techniques are learning a

combination of existing features [12]. Over the last decades, a large number of dimensionality reduction techniques have been proposed; see, e.g., [13, 14] for an overview.

Mapping high-dimensional data to a low-dimensional space is also useful for various reasons, such as reducing the effect of noise, reducing computational cost, and visualizing data. Moreover, due to the curse of dimensionality, classification and regression methods in the original space may not generate satisfactory results and this demonstrates the importance of dimensionality reduction as a pre-processing step. As a result, dimensionality reduction facilitates, among other things, classification, regression, and compression of high-dimensional data.

In the past few years, dimensionality reduction has been an active topic and attracted lots of interest. DR has a wide variety of applications in a number of areas including applied mathematics, computer science, medical research, finance, and engineering. For this reason, quite an amount of work has been done to reduce the dimensionality of the high-dimensional data. We list some of the fields where DR is used:

- Computer vision [15, 16],
- Biomedical informatics [17–19],
- Speech recognition [20, 21],
- Visualization [22, 23],
- Text mining [24, 25],
- Medical [26, 27],
- Agriculture [28, 29],
- Finance [30, 31].

Mathematically speaking, assume that data matrix $X \in \mathbb{R}^{p \times n}$, where p is the number of features and n is the number of observations, has an *"intrinsic"* dimensionality d [32], i.e., the data points lie on or near a manifold with dimension d ($d \ll p$) embedded in the p-dimensional space. This assumption is the base for most of the dimensionality reduction techniques. Most linear DR techniques try to find a projection/transformation matrix $P \in \mathbb{R}^{p \times d}$ by which the data matrix X with dimensionality p is transformed into a new data matrix $Y = P^T X$ with dimensionality d; see Figure 1.3. Often the intrinsic dimensionality d is unknown and there is no assumption on the structure of the low-dimensional manifold.

1.3 Why supervised learning?

Machine learning algorithms, specifically dimensionality reduction techniques, can be classified into three primary categories, *Supervised Learning*, *Unsupervised Learning*, and *Semi-Supervised Learning* depending on having all, no, or some label information. The type of training data (e.g. fully labeled, partially labeled, or unlabeled) used as an input of a machine learning algorithm will distinguish these three categories from each other and they will be defined based on the availability of

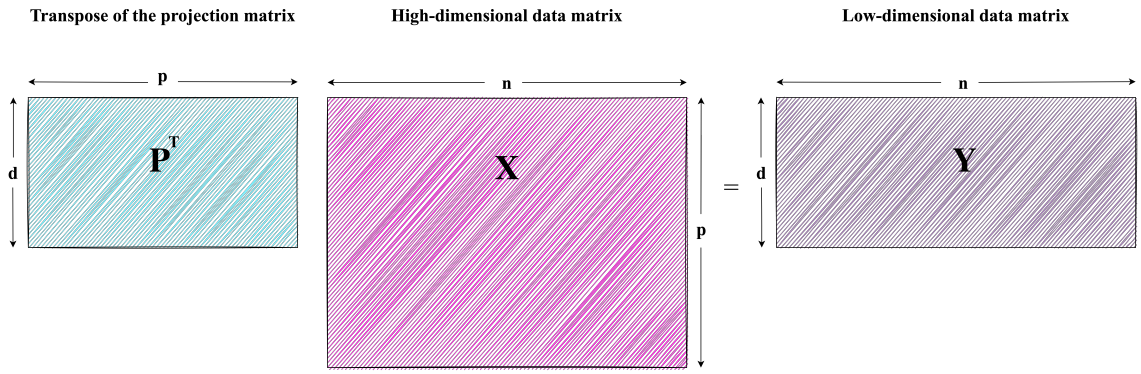


Figure 1.3: Linear dimensionality reduction is the problem of finding the projection matrix P by which the dimension of the high-dimensional data matrix X is reduced.

the output labels of the data. A brief explanation of these categories is given as follows:

- *Supervised Learning*: Fully labeled data points are available and used as training data to train a model that predicts properties of unseen data points. This type of learning has practical applications such as classification and regression.

- *Unsupervised Learning*: Unlabeled data points are available and used as training data to train a model that predicts properties of unseen data points. Since no label is available in this setting, evaluating the performance of a learning algorithm will be difficult and results in having a limited spectrum of applications. Dimensionality reduction and clustering are the most practical applications of unsupervised learning.

- *Semi-Supervised Learning*: Only partial data points are labeled and all data points with or without labels are used as training data to train a model that predicts properties of unseen data points. In this type of learning, a combination of labeled and unlabeled data is employed to train the model. When unlabeled data is easily accessible but labels are expensive to obtain, semi-supervised learning is the best scenario to choose for training the model. Classification, regression, and ranking are practical applications of semi-supervised learning.

Most general dimensionality reduction methods belong to the *unsupervised learning* category which is often much more challenging because there doesn't exist any information about the classes to which data points belong, i.e., data labels are not accessible. On one hand, assessing the results obtained from unsupervised learning methods is hard, since there is no accepted mechanism to validate such results on

an independent data set. So, in unsupervised learning, it is in a sense impossible to check the results, since we do not have access to the "*true*" answer. Instead, fitting a predictive model using a supervised learning technique, one would be able to check the results by seeing how well the response variable is predicted. On the other hand, in most real-world problems, dimensionality reduction is just an intermediate step towards final goals, like classification or regression, in which data labels are playing a significant role to obtain good results. Consider the task of document classification as an example in which feature selection or feature extraction methods are used first to get a low-dimensional text representation, and then, a classifier is trained to make a prediction [33]. Due to the "*lack of supervision*", it is possible that some important words are filtered before training the classifier which affects the final performance [34]. This explains the importance and applicability of supervised learning techniques [35, 36], especially supervised dimensionality reduction methods.

As mentioned earlier, the ultimate goal in many machine learning problems is to efficiently classify high-dimensional data. Classifying the high-dimensional data is still a challenging problem due to the curse of dimensionality. Many features/dimensions are not helpful and may result in critical performance degradation of the classification methods as well as increased computational complexity. To efficiently handle high-dimensional data and to facilitate the classification task, one direct way is to first reduced the dimension of the data by one of the dimensionality reduction methods, and then applying a classification method to classify the embedded data in the low-dimensional space. Although the methodology often works well, we might confront the following challenges while classifying data with high dimensions:

- Powerful dimensionality reduction algorithms are only used to reduce the dimension of the data and are not able to do classification. Therefore, it is hard to

find an appropriate classification method and various classification techniques need to be tried to see which one gives the best result.

- The separability of the data in different classes and consequently the performance of the classification method are not necessarily improved by projecting the data into a low-dimensional space, due to the separation between dimensionality reduction and classification. Since the dimensionality reduction procedure and the subsequent classification technique are employed separately, they don't have a common objective. Thus, it is highly unlikely that we can get optimal results.

One natural way to overcome the above challenges is to integrate dimensionality reduction and classification in a joint framework. In other words, we need to consider the classification requirements together with the dimensionality reduction to make use of the data labels which results in improving the classification performance. Recently, several researchers have dedicated themselves to performing dimensionality reduction and clustering simultaneously [37, 38].

1.4 Major contributions and organization of the dissertation

In this dissertation, we propose a novel supervised dimensionality reduction model that addresses the aforementioned challenges in the previous section. We develop an efficient method that is not only able to reduce the dimension of the high-dimensional data but also to learn a classifier at the same time. Our model is able to solve both dimensionality reduction and classification problems simultaneously.

In Chapter 2, we discuss some of the state-of-the-art methods for dimensionality reduction. We briefly discuss some of the popular dimensionality reduction models

such as Principal Component Analysis (PCA) [39, 40], Linear Discriminant Analysis (LDA) [41, 42], and Supervised Principal Component Analysis (SPCA) [43]. Dual and kernel forms of PCA and SPCA are also described in this chapter. Furthermore, we explore two popular classification methods: Support Vector Machine [44] and k-Nearest Neighbor (kNN) [45]. We also discuss the mathematical backgrounds for solving convex quadratic optimization problems by MATLAB and LIBSVM package [46].

In Chapter 3, we present our novel supervised approach that is able to solve dimensionality reduction and classification problems simultaneously. We develop and discuss the solution procedure for solving the following non-convex optimization problem:

$$\begin{aligned} \min_{P,\alpha} \quad & F(P,\alpha) = \|X - PP^TX\|_F^2 + \mu(\frac{1}{2}\alpha^T A^T PP^T A\alpha - \boldsymbol{e}^T\alpha) \qquad (1.1) \\ \text{s.t.} \quad & P^TP = I_d, \\ & \sum_{i=1}^{n} z_i\alpha_i = 0, \\ & 0 \le \alpha_i \le C, \qquad i = 1, \cdots, n, \end{aligned}$$

where $X = [x_1, x_2, ..., x_n] \in \mathbb{R}^{p\times n}$, $P \in \mathbb{R}^{p\times d}$, $Z = [z_1, z_2, ..., z_n] \in \mathbb{R}^{1\times n}$, $\alpha \in \mathbb{R}^n$, and $\boldsymbol{e} \in \mathbb{R}^n$ are the data matrix, projection matrix, label matrix, the dual variable, and the vector of all ones, respectively, with p being the number of features (dimension of the data), n being the number of observations, and d being the dimension of the low-dimensional space. $A = [z_1x_1, \cdots, z_nx_n] \in \mathbb{R}^{p\times n}$ and $C > 0$ and $\mu > 0$ are hyperparameters in our model that need to be tuned. We also explain the strategy that is used in our model to solve multi-class classification problems. We present our

numerical results and the performance of the model in dimensionality reduction and classification in detail.

Chapter 4 focuses on expanding our model to its nonlinear form by utilizing a nonlinear mapping to handle data sets with nonlinear and complex structures. Classical dimensionality reduction methods often assume that the data points lie on or near a linear low-dimensional space embedded in the original high-dimensional manifold. However, this assumption may not always hold, and when it does not, we are not able to linearly separate the data points. To effectively deal with the nonlinearity in data, we utilize a nonlinear mapping that enables our model to be applied to nonlinear problems. We develop and discuss the solution procedure for solving the following non-convex optimization problem:

$$
\begin{aligned}
\min_{P,\alpha} \quad & F(P,\alpha) = \|\Phi(X) - PP^T\Phi(X)\|_F^2 + \mu(\frac{1}{2}\alpha^T A^T PP^T A\alpha - \boldsymbol{e}^T\alpha) && (1.2)\\
\text{s.t.} \quad & P^T P = I_d, \\
& \sum_{i=1}^{n} z_i\alpha_i = 0, \\
& 0 \le \alpha_i \le C, \qquad\qquad i = 1, \cdots, n,
\end{aligned}
$$

where

$$\Phi : x \to \Phi(x) \in \mathcal{H},$$

is a nonlinear mapping that embeds vector x into some Hilbert space $\mathcal{H}$.
In this chapter, numerical results and the performance of our proposed model in its nonlinear form are presented in detail.

In Chapter 5, we discuss the relevance and importance of our work on solving dimensionality reduction and classification problems. We summarize the conclusions

of the dissertation and some extensions of our work in other possible areas of data science.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter reviews some of the well-known dimensionality reduction methods, including Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Supervised Principal Component Analysis (SPCA). Support Vector Machine (SVM) and k-Nearest Neighbor (kNN) methods will also be described at the end of this chapter.

2.1 Classical dimensionality reduction methods

2.1.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) [39, 40] is one of the most popular unsupervised dimensionality reduction techniques. PCA's goal is to find a low-dimensional linear subspace such that the data points lie on or near this linear manifold. The maximum variability in the data is captured by this low-dimensional space.

"Principal Components" refer to orthogonal vectors that form new coordinates for the linear subspace; see Figure 2.1. The principal components enable us to linearly transform the original data points in a high-dimensional space spanned by the original features into a low-dimensional linear subspace spanned by principal components. PCA has been successfully applied to a wide variety of domains such as face recognition [47], coin classification [48], seismic series analysis [49], visualization [50], noise removal [51], genetics [52], chemistry [53], and physics [54].

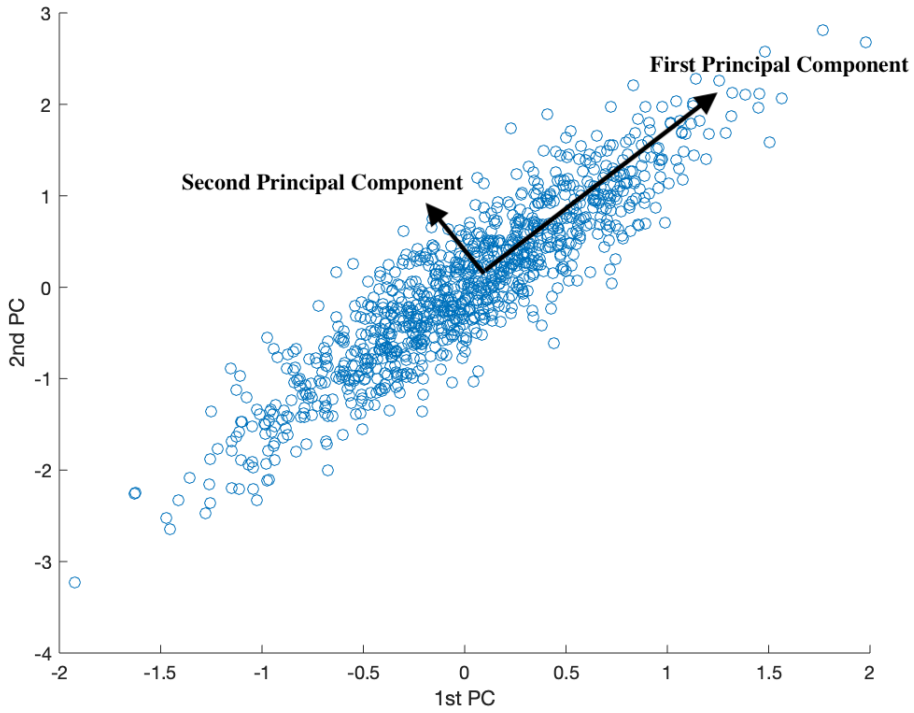


Figure 2.1: A two dimensional projection by PCA applied to a randomly generated data and the new coordinates for the linear subspace.

Assume data vectors x_i $(i = 1, ..., n)$ are packed into data matrix $X \in \mathbb{R}^{p \times n}$. Rows of the data matrix show the features (dimension of the data) and columns are observations. PCA tries to find a linear transformation P that projects the data points into a low-dimensional manifold spanned by principal components, so the number of principal components shouldn't be more than p. The projection onto the linear subspace will minimize the squared reconstruction error [55], i.e., the principal components of a set of data generate the best linear approximations to that data; see Figure 2.2.

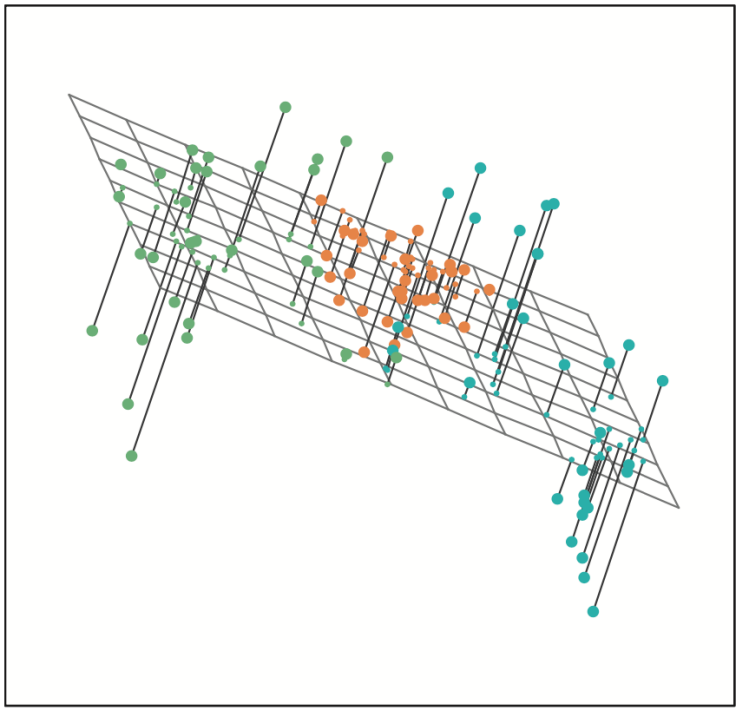


Figure 2.2: Ninety observations simulated in three dimensions. The plane that best fits the data is spanned by the first two principal component directions. The sum of squared distances from each point to the plane is minimized [56, Chapter 10, page 380].

PCA can be described as the following optimization problem:

$$\begin{aligned} \min_{P \in \mathbb{R}^{p \times d}} \quad & \|X - PP^T X\|_F^2 \\ \text{s.t.} \quad & P^T P = I_d. \end{aligned} \tag{2.1}$$

The PCA algorithm is summarized in Algorithm 2.1.

Algorithm 2.1 PCA

Input: Data matrix $X \in \mathbb{R}^{p \times n}$;

Output: Projection matrix $P \in \mathbb{R}^{p \times d}$.

1: Centralize the data: $X = X(I_n - \frac{1}{n}\boldsymbol{e}\boldsymbol{e}^T)$, where $\boldsymbol{e} \in \mathbb{R}^n$ is the vector of all ones;

2: Calculate the covariance matrix of X as:

$$\text{Cov}(X) = XX^T = \sum_{i=1}^{n} x_i x_i^T;$$

3: Find the eigenvalues and eigenvectors of $\text{Cov}(X)$;

4: Sort the eigenvalues in the descending order;

5: Let P be composed of eigenvectors corresponding to the d largest eigenvalues of $\text{Cov}(X)$.

2.1.1.1 Dual PCA

Sometimes the number of features is much larger than the number of observations (i.e., $p \gg n$), and this situation needs to be taken into special consideration when we are working with high-dimensional data, especially for genomics and biology data. In this case, the direct form of PCA (Algorithm 1) is impractical because of computing the eigenvalues and eigenvectors of the very large $p \times p$ covariance matrix. So we need to modify the algorithm such that the run time depends linearly on the number of features p. This will introduce the "Dual" form of PCA. Both PCA and dual PCA find the eigenvalues and eigenvectors of the covariance matrix, however, what makes dual PCA different from PCA is that it performs the eigendecomposition of an $n \times n$ matrix $X^T X$ which results in a significant decrease in the computational cost when $p \gg n$. Dual PCA is summarized in Algorithm 2.2.

Algorithm 2.2 Dual PCA

Input: Data matrix $X \in \mathbb{R}^{p \times n}$;

Output: Projection matrix $P \in \mathbb{R}^{p \times d}$.

1: Centralize the data: $X = X(I_n - \frac{1}{n}\boldsymbol{e}\boldsymbol{e}^T)$, where $\boldsymbol{e} \in \mathbb{R}^n$ is the vector of all ones;

2: Calculate $X^T X$;

3: Find the eigenvalues and eigenvectors of $X^T X$;

4: Sort the eigenvalues in the descending order;

5: Let $V \in \mathbb{R}^{n \times d}$ be composed of eigenvectors corresponding to the d largest eigenvalues of $X^T X$;

6: Let $\Sigma \in \mathbb{R}^{d \times d}$ be a diagonal matrix composed of square roots of the d largest eigenvalues of $X^T X$;

7: $P = XV\Sigma^{-1}$.

2.1.1.2 Kernel PCA

The goal of PCA is to find a low-dimensional linear subspace such that data points lie on or near this low-dimensional space. In many cases, high-dimensional data has a nonlinear structure, and data points lie on or near a nonlinear manifold; see Figure 2.3. In other words, principal components of features are nonlinearly related to the input variables. In this case, PCA is not able to capture the maximum variability in the data. In order to overcome this problem, "Kernel PCA" (KPCA) [57] was proposed. KPCA maps the data to a high-dimensional space where data has sort of a linear structure. This method recasts PCA in a way that it only depends on the inner product of data points, not on the coordinates of the data. KPCA has been successfully applied to different areas such as face recognition [58, 59], speech

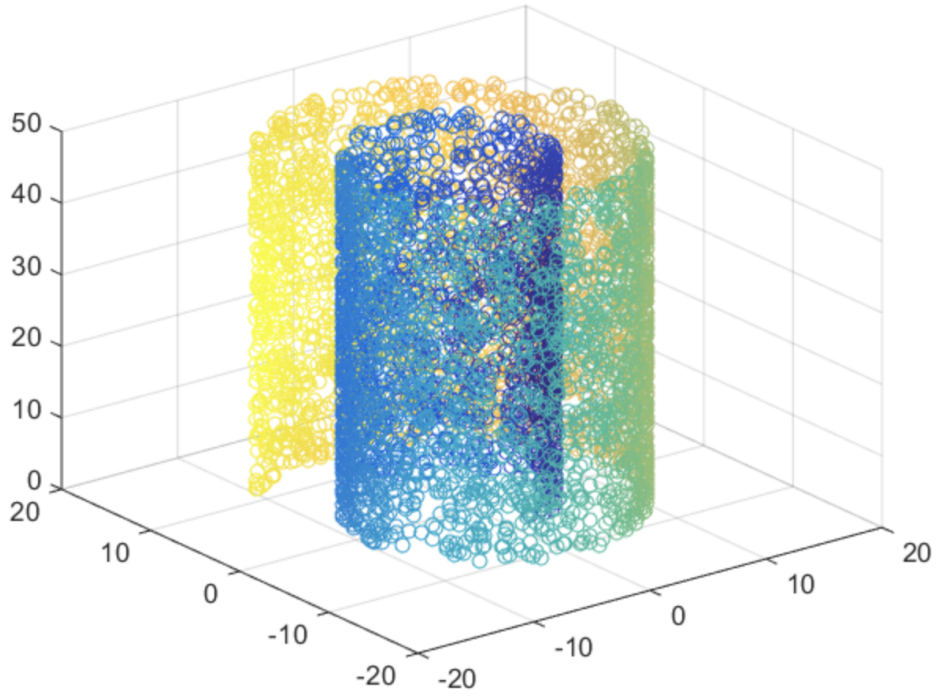


Figure 2.3: 3D Nonlinear Swiss Roll [64].

recognition [60], novelty detection [61], chemical and biological systems [62], and ECG signals [63].

A nonlinear mapping is used to attain the low-dimensional space. Consider the following nonlinear mapping:

$$\Phi : x \to \Phi(x) \in \mathcal{H},$$

where $\mathcal{H}$ is a Hilbert space representing the feature space. Φ also embeds vector x into the space $\mathcal{H}$.

Similar to PCA, the projection onto the manifold will minimize the squared reconstruction error which leads to the following optimization problem for KPCA:

$$\begin{aligned} \min_{P \in \mathbb{R}^{n \times d}} \quad & \|\Phi(X) - PP^T\Phi(X)\|_F^2 \\ \text{s.t.} \quad & P^T P = I_d. \end{aligned} \tag{2.2}$$

To find the projection matrix, first we need to compute the *kernel matrix* by some "*kernel function*". The kernel matrix, denoted by $K(.,.) \in \mathbb{R}^{n \times n}$, only depends on the number of observations, not on the dimension of the feature space. In KPCA, the kernel matrix is introduced to address the inapplicability of PCA and to reduce the dependency on the number of features when the dimension of the data is high. Kernel matrix is a symmetric positive-semidefinite (P.S.D) matrix and every matrix with these properties induces a dot (inner) product defined on some Hilbert space [65]. Therefore, K can be represented as:

$$K = \kappa(X, X) = \langle \Phi(X), \Phi(X) \rangle, \tag{2.3}$$

where κ is a kernel function. Some kernel functions [66] will be explained later. Note that entries of the kernel matrix can also be computed as follows:

$$K_{ij} = \kappa(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle. \tag{2.4}$$

Algorithm 2.3 summarizes the Kernel PCA algorithm.

Algorithm 2.3 Kernel PCA

Input: Data matrix $X \in \mathbb{R}^{p \times n}$ and a kernel function κ;

Output: Projection matrix $P \in \mathbb{R}^{n \times d}$.

1: Calculate the kernel matrix $K_{ij} = \kappa(x_i, x_j)$ by a kernel function;

2: Find the eigenvalues and eigenvectors of K;

3: Sort the eigenvalues in the descending order;

4: Let Σ be a diagonal matrix whose entries are the d largest eigenvalues of K;

5: Let V be a matrix whose columns are eigenvectors corresponding to the d largest eigenvalues of the kernel matrix K;

6: $P = \Phi(X) V \Sigma^{-1}$.

One of the challenges in KPCA is to find an appropriate kernel function for the model [67]. Many different kernels have been proposed by researchers and the most popular ones are:

- Linear kernel:

$$\kappa(x_i, x_j) = x_i^T x_j;$$

- Fisher kernel [68]:

$$\kappa(x_i, x_j) = U_{x_i}^T \mathcal{I}^{-1} U_{x_j},$$

  with U_X being the *Fisher score* and $\mathcal{I}$ being the *Fisher information matrix*;

- Polynomial kernel [69]:

$$\kappa(x_i, x_j) = (\gamma x_i^T x_j + r)^m,$$

  with $r \geq 0$ being a free parameter trading off the influence of higher-order versus lower-order terms in the polynomial. m is the degree of the polynomial and $\gamma > 0$ is the kernel parameter;

- Radial Basis Function kernel (RBF) [70–72]:

$$\kappa(x_i, x_j) = e^{(-\xi \|x_i - x_j\|^2)},$$

  where $\xi > 0$ is the kernel parameter;

- Sigmoid kernel [44] :

$$\kappa(x_i, x_j) = \tanh(\eta x_i^T x_j + s),$$

where η and s are kernel parameters;

- Neural Tangent Kernel (NTK) [73]:

$$\Theta^{(L)}(\theta) = \sum_{p} \partial_{\theta_p} F^{(L)}(\theta) \otimes \partial_{\theta_p} F^{(L)}(\theta),$$

where $F^{(L)} : \mathbb{R}^P \to \mathcal{F}$ is the Artificial Neural Network (ANN) *realization map* that maps parameters θ to functions f_θ in a space $\mathcal{F}$.

Despite of having nice properties and several advantages over classical PCA, KPCA has the following issues:

- The size of the kernel matrix is proportional to n, where n is the number of observations in the data matrix. Thus, as n increases, the size of the kernel matrix also increases. This will result in a high computational cost due to forming K and computing the eigendecomposition. An approach has been proposed to address this problem [74].
- There is no straightforward instruction on how the kernel function is selected. Maximum Variance Unfolding (MVU, known as Semidefinite Embedding) has been proposed to overcome this issue [67].

2.1.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) [41, 42] is one of the most popular supervised dimensionality reduction techniques, and it has applications in many different areas, e.g., bioinformatic [75, 76], biometrics [77, 78], medical [79, 80], and agriculture [81, 82].

Similar to other dimensionality reduction methods, the goal of LDA is to project high-dimensional data onto a low-dimensional manifold. In order to find this low-dimensional space, first two matrices which are called the *between-class variance matrix* and the *within-class variance matrix* are defined. The distance between the means of different classes (separability) and the difference between the mean and the sample data of each class are measured by the between-class variance matrix and within-class variance matrix, respectively. LDA tries to find a low-dimensional manifold such that the between-class variance is maximized and the within-class variance is minimized, thereby maximum class separability will be guaranteed [83, 84]. See e.g., [85, 86] for an overview.

Assume that data vectors x_i $(i = 1, ..., n)$ are packed into data matrix $X \in \mathbb{R}^{p\times n}$, where p is the dimension of the data and n is the number of observations. Let c be the number of classes into which the original data is partitioned and suppose that each class has m_i number of samples. The total mean of the data and the mean of the jth class are denoted by μ and μ_j, respectively. Note that $\sum_{j=1}^{c} m_j = n$. The within-class and between-class variance matrices are defined as follows. The between class variance matrix is:

$$S_b = \sum_{j=1}^{c} m_j S_{b_j},$$

where

$$S_{b_j} = (\mu_j - \mu)(\mu_j - \mu)^T,$$

$$\mu_j = \frac{1}{m_j}\sum_{k=1}^{m_j} x_{jk}, \qquad \mu = \frac{1}{n}\sum_{j=1}^{c}\sum_{k=1}^{m_j} x_{jk}, \qquad j = 1, 2, ..., c,$$

with x_{jk} being the kth data point in the jth class.

The between-class variance matrix is:

$$S_w = \sum_{j=1}^{c} S_{w_j},$$

where

$$S_{w_j} = \sum_{k=1}^{m_j} (x_{jk} - \mu_j)(x_{jk} - \mu_j)^T \quad j = 1, 2, ..., c.$$

LDA seeks a linear transformation W by which maximum linear class separability in the low-dimensional representation is achieved. If this linear transformation is a vector, it maximizes the so-called *Fisher Criterion*:

$$J(w) = \frac{w^T S_b w}{w^T S_w w}.$$

The LDA optimization problem can be formulated as follows:

$$\begin{aligned} \max_{W \in \mathbb{R}^{p \times d}} \quad & \mathbf{Tr}(W^T S_b W) \\ \text{s.t.} \quad & W^T S_w W = I_d. \end{aligned} \tag{2.5}$$

The LDA algorithm is summarized in Algorithm 2.4.

Algorithm 2.4 Linear Discriminant Analysis (LDA)

Input: Data matrix $X \in \mathbb{R}^{p \times n}$;

Output: Projection matrix $W \in \mathbb{R}^{p \times d}$.

1: Find the mean of each class (μ_j) and the total mean of the data (μ);

2: Compute the between-class variance matrix ($S_b \in \mathbb{R}^{p \times p}$) as follows:

$$S_b = \sum_{j=1}^{c} m_j S_{b_j} \qquad \text{where} \qquad S_{b_j} = (\mu_j - \mu)(\mu_j - \mu)^T \qquad j = 1, 2, ..., c;$$

3: Compute the within-class variance matrix ($S_w \in \mathbb{R}^{p \times p}$) as follows:

$$S_w = \sum_{j=1}^{c} S_{w_j} \qquad \text{where} \qquad S_{w_j} = \sum_{k=1}^{m_j} (x_{jk} - \mu_j)(x_{jk} - \mu_j)^T \quad j = 1, 2, ..., c;$$

4: Find the generalized eigenvalues and eigenvectors of (S_b, S_w);

5: Sort the generalized eigenvalues in the descending order;

6: Let W be composed of the eigenvectors corresponding to the d largest generalized eigenvalues.

Despite of being one of the most famous supervised dimensionality reduction techniques, LDA suffers from three major problems:

- If the dimension of the data, p, is higher than the number of observations, n, LDA is not able to find the lower-dimensional manifold. This problem is referred to as the *Small Sample Problem* (SSS) which will result in the singularity of the within-class variance matrix [87–89]. Various approaches have been proposed to address this issue [90–92].
- If data points in different classes are nonlinearly separated, classes will not be discriminated against by LDA. This problem is called the *linearity problem* and there is a solution to this issue in [42].

- LDA is only able to map the data into a $(c-1)$-dimensional space where c is the number of classes.

2.1.3 Supervised Principal Component Analysis (SPCA)

Supervised Principal Component Analysis (SPCA) [43] is another dimensionality reduction method proposed to make use of data labels. It searches for a low-dimensional subspace such that maximum dependency between the projected data and the data labels is achieved. The dependency is measured by Hilbert-Schmidt's Independence Criterion (HSIC) [93]. SPCA is applicable to visualization, classification, and regression problems [43].

Let $X \in \mathbb{R}^{p\times n}$ be the data matrix and $Z \in \mathbb{R}^{l\times n}$ be the matrix of data labels, where l is the number of labels each data point has. According to HSIC, $\mathbf{Tr}(HKHL)$ needs to be maximized, where K is the kernel of P^TX (e.g., X^TPP^TX) with P being the projection matrix, and L is the kernel of Z (e.g., Z^TZ). $H := I_n - \frac{1}{n}\boldsymbol{e}\boldsymbol{e}^T$ is called the *centering matrix*.

SPCA looks for a projection matrix P that projects high-dimensional data into a low-dimensional manifold whose features are uncorrelated. SPCA optimization problem is formulated as follows:

$$\begin{aligned} \max_{P\in\mathbb{R}^{p\times d}} \quad & \mathbf{Tr}(P^TXHLHX^TP) \\ \text{s.t.} \quad & P^TP = I_d. \end{aligned} \tag{2.6}$$

The optimization problem in (2.6) can be considered as an eigenvalue problem and has a closed-form solution. SPCA is summarized in Algorithm 2.5.

Algorithm 2.5 Supervised Principal Component Analysis (SPCA)

Input: Data matrix $X \in \mathbb{R}^{p \times n}$, label matrix $Z \in \mathbb{R}^{l \times n}$, and kernel matrix L;

Output: Projection matrix $P \in \mathbb{R}^{p \times d}$.

1: Let

$$Q = XHLHX^T \qquad \text{where} \quad H = I_n - \frac{1}{n}\boldsymbol{e}\boldsymbol{e}^T;$$

2: Compute the eigenvalues and eigenvectors of Q;

3: Sort the eigenvalues in the descending order;

4: Let P be composed of the eigenvectors corresponding to the d largest eigenvalues of matrix Q.

Note that if the response variable, i.e. data label, is unknown, then set $L = I$. Thus

$$\begin{aligned} XHLHX^T &= XHIHX^T \\ &= (XH)(HX)^T \\ &= [X(I - \frac{1}{n}\mathbf{e}\mathbf{e}^T)][(I - \frac{1}{n}\mathbf{e}\mathbf{e}^T)X]^T \\ &= \text{Cov}(X), \end{aligned} \tag{2.7}$$

where $\text{Cov}(X)$ denotes the covariance matrix of the data matrix X. According to equation (2.7), finding the top d eigenvalues of $XHIHX^T$ is equivalent to finding the d largest eigenvalues of $\text{Cov}(X)$. In other words, PCA is a special case of SPCA with $L = I$.

2.1.3.1 Dual Supervised Principal Component Analysis (Dual SPCA)

Similar to PCA, when the number of features is much larger than the number of observations (i.e., $p \gg n$), SPCA will be impractical because of computing the

eigenvalues and eigenvectors of the very large $p \times p$ matrix Q. Dual Supervised Principal Component Analysis was proposed by Barshan *et al.* [43] to address this problem. The computational cost of dual SPCA has reduced dependence on the number of features, p, so it reduces the dimension of the data with a less computational cost.

Let $Q = XHLHX^T$ where X, L, and H are defined in the previous subsection. Since Q and L are P.S.D matrices, they can be decomposed as follows:

$$L = \Delta^T \Delta$$

$$Q = XHLHX^T = XH\Delta^T \Delta HX^T = \Psi\Psi^T$$

where $\Psi = XH\Delta^T$. Projection matrix P can be computed by Singular Value Decomposition (SVD) [94, 95] of Ψ.

The dual SPCA algorithm is summarized in Algorithm 2.6.

Algorithm 2.6 Dual Supervised Principal Component Analysis (Dual SPCA)

Input: Data matrix $X \in \mathbb{R}^{p \times n}$ and label matrix $Z \in \mathbb{R}^{l \times n}$;

Output: Projection matrix $P \in \mathbb{R}^{p \times d}$.

1: Compute the kernel matrix L for the target variable;

2: Decompose L as

$$L = \Delta^T \Delta;$$

3: Compute matrix Ψ as

$$\Psi = XH\Delta^T \quad \text{where} \quad H = I_n - \frac{1}{n}\boldsymbol{e}\boldsymbol{e}^T;$$

4: Compute the eigenvalues and eigenvectors of

$$\Psi^T\Psi = \Delta H(X^T X)H\Delta^T;$$

5: Sort the eigenvalues in the descending order and let Σ be the diagonal matrix of square roots of the d largest nonzero eigenvalues and V be the eigenvectors associated with the top d eigenvalues;

6: Set $P = \Psi V \Sigma^{-1}$.

2.1.3.2 Kernel Supervised Principal Component Analysis (KSPCA)

In many cases, data with high dimensions has a nonlinear structure, and data points lie on or near a nonlinear manifold; see Figure 2.4. In this case, a nonlinear transformation is required. In order to handle the nonlinearity of the data to which SPCA is applied, Kernel SPCA (KSPCA) [43] was proposed.

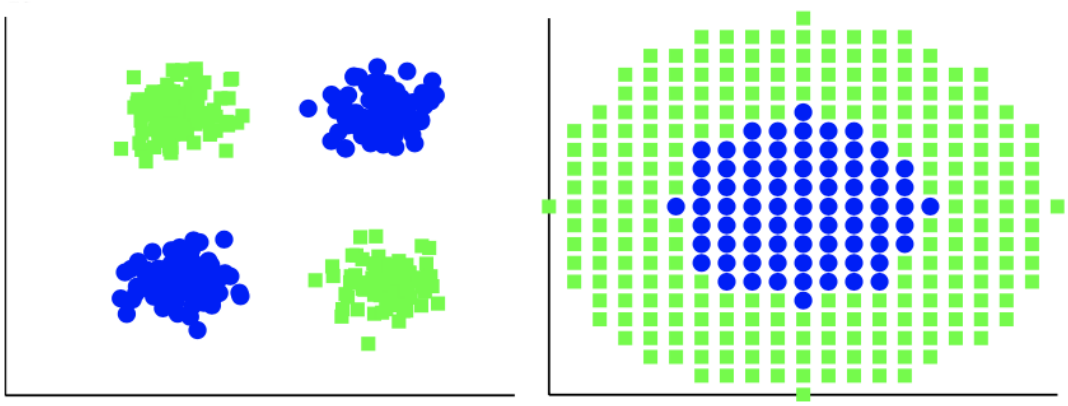


Figure 2.4: The original form of two highly nonlinear artificial data sets (Left: Binary XOR and Right: Concentric Rings) used in data visualization experiments in [43].

Consider the nonlinear mapping Φ:

$$\Phi : x \to \Phi(x) \in \mathcal{H},$$

where $\mathcal{H}$ is a Hilbert space representing the feature space. Φ also embeds vector x into space $\mathcal{H}$.

With this nonlinear mapping, equation (2.6) is changed to:

$$\begin{aligned} \max_{P \in \mathbb{R}^{p \times d}} \quad & \mathbf{Tr}(P^T \Phi(X) H L H \Phi(X)^T P) \\ \text{s.t.} \quad & P^T P = I_d. \end{aligned} \tag{2.8}$$

Transformation matrix P can be expressed as a linear combination of the projected data points [96] as

$$P = \Phi(X)\beta.$$

With this linear combination, problem (2.8) will change to

$$\max_{\beta \in \mathbb{R}^{n \times d}} \quad \mathbf{Tr}(\beta^T K H L H K \beta) \tag{2.9}$$

$$\text{s.t.} \qquad \beta^T K \beta = I_d,$$

where $K \in \mathbb{R}^{n \times n}$ is as in (2.4).

Equation (2.9) is a generalized eigenvalue problem and the solution of this problem, β, is composed of generalized eigenvectors corresponding to the d largest generalized eigenvalues of $(KHLHK, K)$. Kernel SPCA algorithm is summarized in Algorithm 2.7.

Algorithm 2.7 Kernel Supervised Principal Component Analysis (KSPCA)

Input: Kernel matrix for the target variable, L, kernel matrix for training data X_{tr}, $\boldsymbol{K}_{\text{train}} = \Phi(X_{\text{tr}})^T \Phi(X_{\text{tr}})$, kernel matrix for testing data X_{te}, and $\boldsymbol{K}_{\text{test}} = \Phi(X_{\text{tr}})^T \Phi(X_{\text{te}})$;

Output: Dimension reduced training and testing data, Y_{tr} and Y_{te}.

1: Compute matrix Q as

$$Q = \boldsymbol{K}_{\text{train}} HLH \boldsymbol{K}_{\text{train}} \qquad \text{where} \qquad H = I - \frac{1}{n} \mathbf{e}\mathbf{e}^T;$$

2: Compute the generalized eigenvalues and generalized eigenvectors of $(Q, \boldsymbol{K}_{\text{train}})$;

3: Sort the generalized eigenvalues in the descending order and let β be composed of the generalized eigenvectors corresponding to the d largest generalized eigenvalues.

4: Reduce the dimension of the training data X_{tr}

$$Y_{\text{tr}} = \beta^T [\Phi(X_{\text{tr}})^T \Phi(X_{\text{tr}})] = \beta^T \boldsymbol{K}_{\text{train}};$$

5: Reduce the dimension of the testing data X_{te}

$$Y_{\text{te}} = \beta^T [\Phi(X_{\text{tr}})^T \Phi(X_{\text{te}})] = \beta^T \boldsymbol{K}_{\text{test}}.$$

2.2 Classification methods

2.2.1 Support Vector Machine (SVM)

Support Vector Machine (SVM), proposed by Vapnik [44], is one of the earliest supervised learning methods for classification. With a solid theoretical background and a geometrical interpretation, SVM tries to separate the data points with a large "*gap*". That is, a classifier hyperplane of a binary labeled data set is found by SVM such that maximum margin between positive and negative samples is achieved; see Figure 2.5. Several efficient and fast SVM packages such as LIBSVM [46, 97] and SVM$^{\text{light}}$ [98] have been developed since the 1990s. SVM was initially developed for binary classification and has since been extended for multi-class classification problems [99, 100]. Support Vector Machine has been successfully applied and used in different areas such as pattern recognition [101], computational biology (medical) [102], and image processing [103].

Assume that training data vectors x_i $(i = 1, ..., n)$ are packed into data matrix $X \in \mathbb{R}^{p\times n}$ and $z \in \mathbb{R}^n$ is a vector of the data labels with $z_i \in \{1, -1\}$ $(i = 1, ..., n)$. SVM can be formulated as a constrained optimization problem and its objective function can be derived using its geometrical interpretation.

Hard-margin SVM classification: The goal of SVM is to find a separating hyperplane $w^T x + b = 0$ by which the maximum margin between positive and negative classes is guaranteed. Vapnik's Structural Risk Minimization explains the reason behind maximizing the margin [44]. To find this hyperplane, first of all, we assume that data is noise-free and can be linearly classified by a hyperplane. This is how the *hard-margin* SVM is introduced. The hard-margin SVM can be formulated as the following constrained optimization problem:

$$\min_{w\in\mathbb{R}^p, b\in\mathbb{R}} \qquad \frac{1}{2} w^T w \tag{2.10}$$

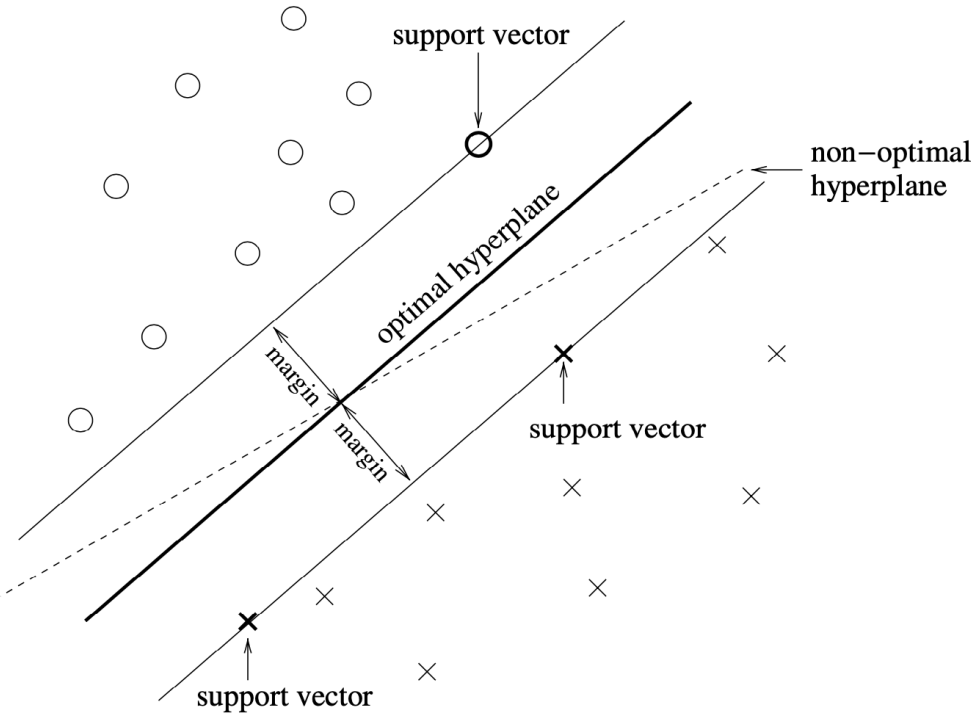


Figure 2.5: Optimal hyperplane found by SVM applied to a binary class classification problem [104].

$$\text{s.t.} \qquad z_i(w^T x_i + b) \geq 1, \qquad i = 1, 2, ..., n.$$

Soft-margin SVM classification: In many cases, the data is noisy and it is not linearly separable. In other words, no (w, b) satisfies the constraints of (2.10). In this situation, problem (2.10) will become "infeasible". To overcome this issue, *soft-margin* SVM was introduced to achieve maximum margin at a cost of having a few misclassified data points. That is, soft-margin SVM maximizes the margin and minimizes the degree of total violations simultaneously. A slack variable ϵ_i $(i = 1, 2, ..., n)$ is introduced which measures the degree of misclassification. The following is the constrained optimization problem of the soft-margin SVM:

$$\min_{w,b,\epsilon} \qquad \frac{1}{2} w^T w + C \sum_{i=1}^{n} \epsilon_i \tag{2.11}$$

$$\begin{aligned} \text{s.t.} \qquad & z_i(w^T x_i + b) \geq 1 - \epsilon_i, \qquad && i = 1, 2, ..., n, \\ & \epsilon_i \geq 0, && i = 1, 2, ..., n, \end{aligned}$$

where $C > 0$ is the regularization parameter that controls the sum of the ϵ_i's. So the severity of the violations to the margin are determined by C. When C is small, one would seek narrow margins that are rarely violated. On the other hand, when C is large, the margin is wider and more violations to it is allowed; see Figure 2.6. Lin [105] theoretically proved that for large C and linearly separable data, problem (2.11) goes back to problem (2.10) with all ϵ_i being zero.

The dimension of vector w is the same as the dimension of the data points and due to the possible high dimensionality of the data, w can be of high dimension. To handle this difficulty, people usually solve the *dual* form of soft-margin SVM by employing the *Lagrange function*. The dual problem can be formulated as follows:

$$\begin{aligned} \min_{\alpha \in \mathbb{R}^n} \qquad & \frac{1}{2}\alpha^T Q \alpha - \boldsymbol{e}^T \alpha \qquad && (2.12) \\ \text{s.t.} \qquad & \sum_{i=1}^{n} z_i \alpha_i = 0, \\ & 0 \leq \alpha_i \leq C, \qquad && i = 1, 2, ..., n, \end{aligned}$$

where α is the dual vector and $Q \in \mathbb{R}^{n \times n}$ is a positive semi-definite matrix whose entries are calculated by:

$$Q_{ij} = z_i z_j {x_i}^T x_j$$

with z_i being the label of the ith data point.

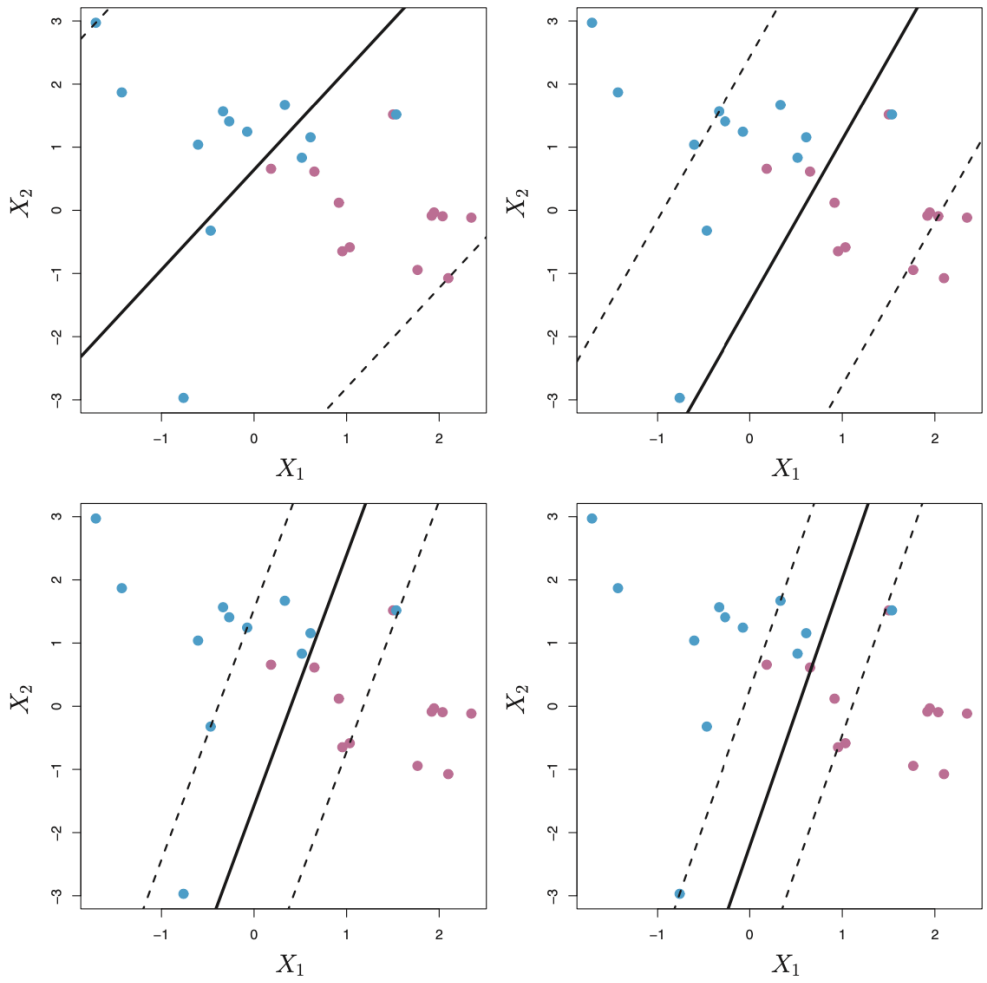


Figure 2.6: A support vector classifier was fit using four different values of the tuning parameter C in (2.11). The largest value of C was used in the top left panel, and smaller values were used in the top right, bottom left, and bottom right panels. When C is large, there is a high tolerance for observations being on the wrong side of the margin, and so the margin will be large. As C decreases, the tolerance for observations being on the wrong side of the margin decreases, and the margin narrows [56, Chapter 9, page 348].

Large scale quadratic programming such as MATLAB built-in function quadprog or LIBSVM package [46, 97] can be employed to solve the optimization problem in (2.12). The procedure of how the convex quadratic optimization problem in (2.12) is solved by quadprog and LIBSVM will be explained at the end of this chapter. After

solving the optimization problem (2.12), with the help of the **Karush-Kuhn-Tucker (KKT)** optimality conditions of problems (2.11) and (2.12), the normal vector of the hyperplane, w, and the y-intercept of the hyperplane, b, can be calculated by:

$$w = \sum_{i=1}^{n} z_i \alpha_i x_i$$

and

$$b = -\frac{\sum_{i:0<\alpha_i<C} z_i(Q\alpha_i - \boldsymbol{e})}{|\{i : 0 < \alpha_i < C\}|},$$

and the label of the new data $\boldsymbol{x}$ (testing data) can be found by the following *decision function*:

$$DF(\boldsymbol{x}) = \text{sgn}(w^T\boldsymbol{x} + b) = \text{sgn}\left(\sum_{i=1}^{n} z_i \alpha_i x_i^T \boldsymbol{x} + b\right),$$

with sgn being the sign function.

2.2.1.1 Kernel SVM

Most often, data has a highly nonlinear distribution and fitting only by a linear classifier may cause many training misclassifications. So, underfitting will occur and the decision function will have a poor performance. Another trick to deal with nonlinearly separable data is to use the *kernel trick* which will extend linear SVM to nonlinear SVM. This problem is referred to as *kernel* SVM classification. The main issue here is, modeling a nonlinear curve is a frustrating task and we are only familiar with parabolic, hyperbolic, or elliptic curves which, in practice, are far from being enough. So, mapping data to a high-dimensional space is another approach that can be taken into consideration. Data is more likely linearly separable in this

high-dimensional space with higher possibility [106]. So in this method, first the nonlinear mapping Φ transforms the input data to a high-dimensional space where the data is more likely linearly separable. Then a hyperplane of maximal margin is found by SVM in the manifold with new features. The followings are the primal and dual forms of kernel SVM classification problem:

- Primal:

$$\begin{aligned} \min_{w,b,\epsilon} \quad & \frac{1}{2}w^T w + C\sum_{i=1}^{n}\epsilon_i & \\ \text{s.t.} \quad & z_i(w^T\Phi(x_i)+b) \geq 1-\epsilon_i, & i=1,2,...,n, \\ & \epsilon_i \geq 0, & i=1,2,...,n, \end{aligned} \tag{2.13}$$

- Dual:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T\tilde{Q}\alpha - \boldsymbol{e}^T\alpha & \\ \text{s.t.} \quad & \sum_{i=1}^{n} z_i\alpha_i = 0, & \\ & 0 \leq \alpha_i \leq C, & i=1,2,...,n, \end{aligned} \tag{2.14}$$

where $\tilde{Q}_{ij} = z_i z_j \kappa(x_i, x_j)$. The normal vector w, y-intercept b, and the label of the testing data $\boldsymbol{x}$ are respectively calculated by:

$$w = \sum_{i=1}^{n} z_i\alpha_i\Phi(x_i),$$

$$b = -\frac{\sum_{i:0<\alpha_i<C} z_i(\tilde{Q}\alpha_i - \boldsymbol{e})}{|\{i : 0<\alpha_i<C\}|},$$

$$DF(\boldsymbol{x}) = \text{sgn}(w^T\Phi(\boldsymbol{x})+b) = \text{sgn}\left(\sum_{i=1}^{n} z_i\alpha_i\kappa(x_i,\boldsymbol{x})+b\right).$$

2.2.2 k-Nearest Neighbor (kNN)

k-Nearest Neighbor (kNN) [45] is a supervised learning method used for classification. The idea of kNN is to assign the same class (label) to observations that are *"similar"* to each other. In other words, kNN tries to find the k closest (most similar) training points according to some criterion, where k is a positive integer. However, deciding whether the two data points are similar or not is quite an open question. So, in order to find similar data points, we need to find a way to compare them. However, data might be of many different types, such as number, true/false (boolean), or categorical, which makes comparison difficult. The most common solution to this problem is to convert all characteristics into numerical values. Once everything is converted into numbers, we can compare the observations and find similar ones. kNN uses some *"distance"* between data points, under some appropriate metric, to interpret their similarity. Euclidean, Chebychev, Minkowski, Hamming, and Mahalanobis are among the most popular distances that can be used to find similar observations. This method is usually used when the data set is small and noise-free. kNN is successfully applied in a variety of applications such as data compression and forecasting.

Suppose training data vectors x_i $(i = 1, ..., n)$ are packed into data matrix $X \in \mathbb{R}^{p\times n}$. Assume that $z \in \mathbb{R}^n$ is a vector consisting of the data labels. For simplicity, we consider the binary classification in which the data set has only two classes, i.e., $\forall i, z_i \in \{1, -1\}$. Given the training data and its labels, the distances between the data points in the training set are calculated to find similar observations and assign the same label (class) to them, i.e., for each data point x_i we calculate:

$$\text{dist}(x_i, x_j) \qquad\qquad \forall\ i, j = 1, ..., n,$$

where dist stands for the distance that is used.

One should note that there are two important concepts that need to be taken into consideration: (1) the method to calculate the distances between the data points and (2) the parameter k to decide how many neighbors will be chosen for the kNN model. The latter will be discussed in this section. Once appropriate metric and parameter k are chosen, to classify a new observation, the distance from this new data point and all other points in the training set is calculated, then k closest training points will be called. kNN algorithm boils down to forming a majority vote between the k most similar instances to the given "unseen" observation. Simply, the label of the unseen data point will be determined by the labels of the majority of the similar observations to this new data point. An odd k is usually chosen to prevent tie situations.

The very first question that might be asked is: *How to choose k and how it affects the classifier*? In the kNN algorithm, k is being considered as a hyperparameter that needs to be tuned in order to get the best result. The appropriate choice of k will have a significant impact on the performance of kNN. k can be thought of as a parameter that controls the shape of the decision boundary. By choosing small k, the region will be forced to be more flexible which will result in having less bias but the high variance and being less stable. On the other hand, higher values of k lead to having smoother decision boundaries which will reduce variance caused by random error but increase bias; see Figure 2.7. Picking large k will also make the model more resilient to outliers due to taking the average of more votes. k is usually set to be equal to the square root of the number of observations in the training set.

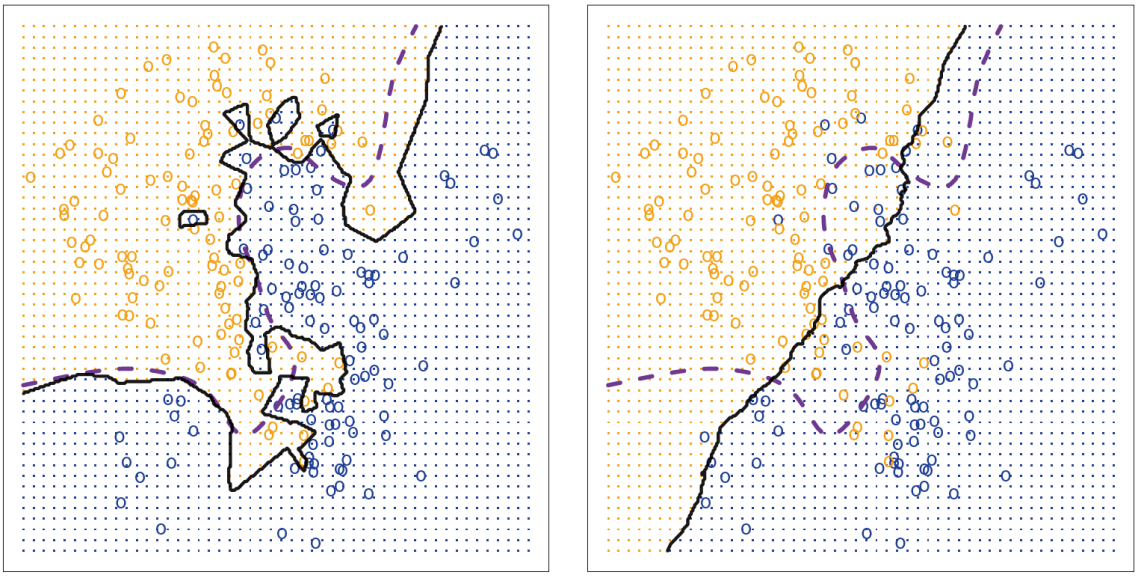


Figure 2.7: A comparison of the kNN decision boundaries (solid black curves) obtained using $k = 1$ and $k = 100$ on a simulated data set consisting of 100 observations. With $k = 1$, the decision boundary is overly flexible (Left). With $k = 100$ it is not sufficiently flexible (Right). The Bayes decision boundary is shown as a purple dashed line [56, Chapter 2, page 41].

2.3 Solving the convex quadratic problem in the dual form of SVM

2.3.1 Solving the convex quadratic problem by LIBSVM

Consider the following convex quadratic optimization problem:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T Q\alpha - \boldsymbol{e}^T\alpha \\ \text{s.t.} \quad & \sum_{i=1}^{n} z_i\alpha_i = 0, \\ & 0 \le \alpha_i \le C, \qquad i = 1, \cdots, n, \end{aligned} \tag{2.15}$$

where $\boldsymbol{e}$ is the vector of all ones, C is the upper bound of all variables, and Q is a P.S.D matrix of size $n \times n$. One should note that if Radial Basis Function (RBF) kernel $\kappa(x_i, x_j) = e^{(-\xi\|x_i - x_j\|^2)}$ (with $x_i \neq x_j \ \forall i, j$) is used, then Q will be positive definite (P.D). Problem (2.15) can be solved by LIBSVM [46, 97].

One of the difficulties we usually face is that Q is usually dense (i.e., all Q's components are non-zero) and may be too large to be stored. In LIBSVM, a decomposition method is considered to address this difficulty in solving the dual convex quadratic problem. Using this decomposition method, only a subset of α is modified and updated in each iteration, so we only need some columns of Q to do the operation. This small subset, the working set B, leads to a smaller optimization subproblem. Sequential Minimal Optimization (SMO) [107] is an extreme case of the decomposition methods used in LIBSVM [97]. An SMO-type decomposition method [108] is considered in LIBSVM.

The decomposition method used in LIBSVM to solve the convex quadratic optimization problem (2.15) is an iterative method that generates a sequence of approximations for which the objective function is monotonically decreasing. This sequence of approximations is always convergent. For more details on the convergence of this method the reader can see, e.g., [109, 110].

2.3.2 Solving the convex quadratic problem by MATLAB built-in function quadprog

Consider the following convex quadratic problem:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2}x^T Hx + c^T x \\ \text{s.t.} \quad & Ax \leq b, \\ & A_{eq}x = b_{eq}, \\ & l \leq x \leq u, \end{aligned} \tag{2.16}$$

where $H \in \mathbb{R}^{n \times n}$ is the *Hessian matrix* of the objective function in (2.16) which needs to be positive-semi definite. Note that H can be a "sparse" or a "full" matrix. The *interior-point-convex* algorithm [111, Chapter 11] is utilized by the MATLAB built-in function quadprog to solve the optimization problem (2.16).

CHAPTER 3

A NOVEL SUPERVISED DIMENSIONALITY REDUCTION METHOD

3.1 Introduction

Dimensionality reduction is a necessity before classification when we are facing high-dimensional data. To facilitate the classification task of high-dimensional data, first the dimension of the data should be reduced by one of the dimensionality reduction methods, and then the reduced dimension data is classified by one of the classification techniques. However, on one hand, most of the powerful dimensionality reduction methods are not able to do classification. Therefore, after reducing the dimension of the data, finding an appropriate classification method is hard and time-consuming and several methods need to be tried to obtain the best result. On the other hand, performing dimensionality reduction on input high-dimensional data samples to obtain the low-dimensional representation and successively classifying them by means of a typical classifier is a stepwise manner that may overlook the dependency between the two processes, resulting in compromise of classification accuracy. In other words, two objectives independent of dimensionality reduction and classification are done separately, and thus, achieving optimal results is not guaranteed.

In this study, we propose a novel supervised dimensionality reduction method that integrates PCA with the dual form of SVM. We elegantly unify the two processes, namely dimensionality reduction and classification, by formulating a novel constrained optimization model, which is supervised and takes advantage of the label information. Our model simultaneously finds low-dimensional representations and a classifier

hyperplane. Moreover, we propose an alternating iteration algorithm to solve the constrained optimization problem of our proposed model.

High accuracy and effective representation are the advantages of our supervised model since it is learning the low-dimensional representation and the classifier simultaneously. Moreover, since our model utilizes the data labels, it is guaranteed that in the reduced space, the data points from the same class are as close as possible to each other and this will result in having a higher classification accuracy. Our model is also able to solve multi-class classification problems. Our proposed model is formulated as follows:

$$\begin{aligned} \min_{P,\alpha} \quad & F(P,\alpha) = \|X - PP^TX\|_F^2 + \mu(\frac{1}{2}\alpha^T A^T PP^T A\alpha - \boldsymbol{e}^T\alpha) \qquad (3.1) \\ \text{s.t.} \quad & P^TP = I_d, \\ & \sum_{i=1}^{n} z_i\alpha_i = 0, \\ & 0 \le \alpha_i \le C, \qquad i = 1, \cdots, n, \end{aligned}$$

where $X = [x_1, x_2, ..., x_n] \in \mathbb{R}^{p\times n}$, $P \in \mathbb{R}^{p\times d}$, $Z = [z_1, z_2, ..., z_n] \in \mathbb{R}^{1\times n}$, $\alpha \in \mathbb{R}^n$, and $\boldsymbol{e} \in \mathbb{R}^n$ are the data matrix, projection matrix, label matrix, the dual variable, and the vector of all ones, respectively, with p being the number of features (dimension of the data), n being the number of observations, and d being the dimension of the low-dimensional space. $A = [z_1x_1, \cdots, z_nx_n] \in \mathbb{R}^{p\times n}$ and $C > 0$ and $\mu > 0$ are hyperparameters in our model that need to be tuned. Data matrix X and label matrix Z are also known. The goal of our model is to find the projection matrix P being used in dimensionality reduction and to find the classifier hyperplane $w^TP^Tx + b = 0$ being used for data classification. With the help of the dual variable α, w and b which

are the normal vector and y-intercept of the separating hyperplane, respectively, can be obtained.

Problem (3.1) is non-convex, but it can be solved in an alternating way as follows:

- α-subproblem (fixing P)

$$\begin{aligned} \min_{\alpha\in\mathbb{R}^n} \quad & f(\alpha) = \frac{1}{2}\alpha^T A^T P P^T A\alpha - \boldsymbol{e}^T\alpha \\ \text{s.t.} \quad & \sum_{i=1}^{n} z_i\alpha_i = 0, \\ & 0 \le \alpha_i \le C, \qquad i = 1, \cdots, n; \end{aligned} \tag{3.2}$$

- P-subproblem (fixing α)

$$\begin{aligned} \min_{P\in\mathbb{R}^{p\times d}} \quad & g(P) = \|X - PP^T X\|_F^2 + \frac{\mu}{2}\alpha^T A^T P P^T A\alpha \\ \text{s.t.} \quad & P^T P = I_d. \end{aligned} \tag{3.3}$$

Problem (3.3) can be converted into a trace optimization problem. In fact,

$$\begin{aligned} g(P) :&= \|X - PP^T X\|_F^2 + \frac{\mu}{2}\alpha^T A^T P P^T A\alpha \\ &= \|X\|_F^2 - \|P^T X\|_F^2 + \frac{\mu}{2}\|P^T A\alpha\|_2^2 \\ &= \mathbf{Tr}(X^T X) - \mathbf{Tr}(P^T X X^T P) + \frac{\mu}{2}\mathbf{Tr}(P^T A\alpha\alpha^T A^T P) \\ &= \mathbf{Tr}(P^T T P) + \mathbf{Tr}(X^T X), \end{aligned} \tag{3.4}$$

where $T = \frac{\mu}{2}A\alpha\alpha^T A^T - XX^T$. Hence, the P-subproblem (3.3) is equivalent to:

- P-subproblem (fixing α)

$$\begin{aligned} \min_{P \in \mathbb{R}^{p \times d}} \quad & g(P) = \mathbf{Tr}(P^T T P) \\ \text{s.t.} \quad & P^T P = I_d. \end{aligned} \tag{3.5}$$

3.1.1 Solving subproblems (3.2) and (3.5)

The optimization problem (3.2) is a convex quadratic optimization problem that can be solved by a large scale quadratic programming solver such as MATLAB built-in function quadprog or LIBSVM package [46, 97]. Problem (3.5) is equivalent to finding the smallest d eigenvalues of matrix T [112]. We can solve problem (3.2) for α and problem (3.5) for P, alternatingly until convergence.

The outputs of our alternating algorithm are the projection matrix P and the dual variable α. P is directly utilized to reduce the dimension of the data and α helps to find the classifier hyperplane. As discussed in Chapter 2, with the help of the KKT optimality conditions of the SVM optimization problem, after solving (3.2) and obtaining α, the normal vector, w, and the y-intercept, b, of the classifier hyperplane can be calculated by:

$$w = \sum_{i=1}^{n} \alpha_i z_i P^T x_i \tag{3.6}$$

and

$$b = -\frac{\sum_{i:0<\alpha_i<C} z_i \nabla f(\alpha)_i}{|\{i : 0 < \alpha_i < C\}|}, \tag{3.7}$$

where $\nabla f(\alpha) = (A^T P P^T A)\alpha - \boldsymbol{e}$ is the gradient of the objective function in (3.2).

Once all variables are found, the label of the new and unseen data $\boldsymbol{x}$ (testing data) can be predicted by the following decision function:

$$DF(\boldsymbol{x}) = \text{sgn}(w^T P^T \boldsymbol{x} + b) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i z_i x_i^T P P^T \boldsymbol{x} + b\right), \tag{3.8}$$

where sgn is the sign function.

Our proposed supervised algorithm is summarized in Algorithm 3.1.

Algorithm 3.1 Linear Supervised PCA-SVM

Input: $X, P^0, Z, \mu > 0, C > 0, d$, tolerance, and $k = 1$;

Output: $P \in \mathbb{R}^{p\times d}, \alpha \in \mathbb{R}^n$.

construct matrix $A = [z_1 x_1, \cdots, z_n x_n]$;

while $|F(P^{k+1}, \alpha^{k+1}) - F(P^k, \alpha^k)| \geq$ tolerance **do**

  for given P^k, solve α-subproblem (3.2) for α^k;

  $T^k = \frac{\mu}{2} A(\alpha^k)(\alpha^k)^T A^T - XX^T$;

  find eigenvalues and eigenvectors of T^k;

  sort eigenvalues in the ascending order;

  let P^k be composed of the eigenvectors corresponding to the d smallest eigenvalues of T^k;

  $k \leftarrow k + 1$;

end while

3.2 Convergence of the proposed algorithm

In this section, we study and discuss the convergence of our proposed algorithm in Algorithm 3.1. To solve large-scale optimization problems, the big original problem is usually broken down into small subproblems and then alternating methods are employed to solve these small subproblems and finally get the optimal solution of the

original one. In other words, the original intractable problem is transformed into a set of tractable subproblems by alternating procedures.

In our model, α and P are updated and optimized alternatingly. In our experiments, LIBSVM [46,97] and MATLAB function quadprog are employed to solve the convex quadratic optimization problem (3.2). The procedure of how LIBSVM and quadprog solve this problem and their convergence is discussed in Chapter 2. In this section, we discuss and explore the convergence of the trace optimization problem (3.5), and then comment on the convergence of Algorithm 3.1 for the original optimization problem and prove some nice properties of our model.

The trace optimization problem (3.5) yields an eigenvalue problem [112]. In (3.5), given a symmetric matrix T of size $p \times p$, the trace of $P^T TP$ is minimized when P is an orthogonal basis of the eigenspace associated with the (algebraically) smallest eigenvalues. If eigenvalues are labeled in the increasing order and $v_1, ..., v_d$ are eigenvectors associated with the first d eigenvalues $\lambda_1, ..., \lambda_d$, $P = [v_1, ..., v_d]$, with $P^T P = I_d$, is the minimizer and,

$$\mathbf{Tr}[P^T TP] = \lambda_1 + ... + \lambda_d. \tag{3.9}$$

The conclusion is an immediate consequence of the Courant–Fisher characterization [113, 114]. Any P which is an orthonormal basis of the eigenspace associated with the d smallest eigenvalues is the optimal solution of (3.5) [115, 116]. Note that the optimal solution P of the problem (3.5) is not unique, however, what matters the most is the subspace constructed by any orthonormal basis rather than a particular orthonormal basis itself.

In the following we discuss the convergence of our proposed algorithm. Suppose $\{\alpha^k\}_{k=1}^{\infty}$ and $\{P^k\}_{k=1}^{\infty}$ are the sequences generated by Algorithm 3.1, i.e., suppose α^k and P^k are the optimal (exact) solutions of (3.2) and (3.5), respectively, in the kth iteration. We have the following results:

Theorem 3.2.1. *Suppose $\{\alpha^k\}_{k=1}^{\infty}$ and $\{P^k\}_{k=1}^{\infty}$ are the sequences generated by Algorithm 3.1.*

(i) The sequence $\{\alpha^k\}_{k=1}^{\infty}$ has a convergent subsequence.

(ii) The sequence $\{P^k\}_{k=1}^{\infty}$ has a convergent subsequence.

Proof. ***(i)*** The inequality constraint in (3.2) is a box constraint. Let $\mathcal{D} = [0, C]^n$. $\mathcal{D}$ is a bounded, closed, and convex set, so it is compact. By **Bolzano–Weierstrass** theorem, the sequence $\{\alpha^k\}_{k=1}^{\infty}$ contains a convergent subsequence. This means there exists $\tilde{\alpha}^* \in \mathcal{D}$ to which a subsequence of $\{\alpha^k\}_{k=1}^{\infty}$, denoted by $\{\alpha^{k_j}\}_{j=1}^{\infty}$, converges to $\tilde{\alpha}^*$, i.e.,

$$\lim_{j\to\infty} \alpha^{k_j} = \tilde{\alpha}^*.$$

(ii) Let $\mathcal{O} = \{M : M \in \mathbb{R}^{p\times d} \ \& \ M^T M = I_d\}$ be the set of orthonormal matrices in $\mathbb{R}^{p\times d}$. $\mathcal{O}$ is a closed and bounded subset (as a subset of $\mathbb{R}^{pd}$), so it is compact. By **Bolzano–Weierstrass** theorem, the sequence $\{P^k\}_{k=1}^{\infty}$ contains a convergent subsequence. So, there exists $\tilde{P}^* \in \mathcal{O}$ to which a subsequence of $\{P^k\}_{k=1}^{\infty}$, denoted by $\{P^{k_i}\}_{i=1}^{\infty}$, converges to $\tilde{P}^*$, i.e.,

$$\lim_{i\to\infty} P^{k_i} = \tilde{P}^*.$$

□

In the following, we show that the limit points of subsequences $\{\alpha^{k_j}\}_{j=1}^{\infty}$ and $\{P^{k_i}\}_{i=1}^{\infty}$ satisfy the KKT optimality conditions of problems (3.2) and (3.5), respectively. The KKT conditions for these subproblems are as follows:

- The KKT conditions of (3.2):
  - Feasibility conditions:

$$\sum_{i=1}^{n} \alpha_i z_i = 0, \tag{3.10}$$

$$\alpha_i - C \leq 0, \qquad i = 1, ..., n, \tag{3.11}$$

$$-\alpha_i \leq 0, \qquad i = 1, ..., n, \tag{3.12}$$

$$\lambda_i \geq 0, \qquad i = 1, ..., n, \tag{3.13}$$

$$\eta_i \geq 0, \qquad i = 1, ..., n; \tag{3.14}$$

  - Complementarity conditions:

$$\lambda_i \alpha_i = 0, \qquad i = 1, ..., n, \tag{3.15}$$

$$\eta_i(\alpha_i - C) = 0, \qquad i = 1, ..., n; \tag{3.16}$$

  - First order condition:

$$A^T P P^T A\alpha - \boldsymbol{e} + \beta \boldsymbol{z} - \boldsymbol{\lambda} + \boldsymbol{\eta} = 0, \tag{3.17}$$

    where $\boldsymbol{z}, \boldsymbol{\lambda}$ and $\boldsymbol{\eta}$ are the label vector and dual variables, respectively.

- The KKT conditions of (3.5):
  - Feasibility condition:

$$P^T P = I_d; \tag{3.18}$$

– First order condition:

$$(\mu A\alpha\alpha^T A^T - XX^T)P + 2P\bar{R} = 0, \tag{3.19}$$

where $\bar{R}$ is the dual variable of the problem (3.5).

Since $\forall k$, α^k and P^k are optimal solutions of problems (3.2) and (3.5) in the kth iteration, they satisfy the KKT optimality conditions of these problems. Consequently, $\{\alpha^{k_j}\}_{j=1}^{\infty}$ and $\{P^{k_i}\}_{i=1}^{\infty}$ also satisfy the KKT conditions of these two subproblems. By taking the limit of all the above equations and letting $j \to \infty$ and $i \to \infty$, it can be easily seen that the KKT conditions of problems (3.2) and (3.5) are also met by the limit points $\tilde{\alpha}^*$ and $\tilde{P}^*$. This discussion leads to the following conclusion:

Conclusion 1: The KKT optimality conditions of optimization problems (3.2) and (3.5) are met by the limit points $\tilde{\alpha}^*$ and $\tilde{P}^*$.

We also show that the objective function in (3.1) is decreasing and convergent.

Theorem 3.2.2. *Let $\{\alpha^k\}$ and $\{P^k\}$ be generated by Algorithm 3.1. Then the sequence $\{F(P^k, \alpha^k)\}_{k=1}^{\infty}$ in (3.1) is decreasing and convergent.*

Proof. For any k, let $M^k = A^T P^k P^{kT} A$ and $l = \min\{-\boldsymbol{e}^{\mathrm{T}}\alpha^{\mathrm{k}} : \forall \mathrm{k}\}$. Suppose $0 < C < \infty$, where C is the hyperparameter of the model. Since α^k has n components and for each of its components, $0 \leq {\alpha_i}^k \leq C$, $\forall i, k$, l is a finite number and lies in the closed interval $[-nC, 0]$, where n is the number of observations. Moreover, since $\forall k$, M^k is P.S.D, we have:

$$\mu l \leq \|X - (P^k)(P^k)^T X\|_F^2 + \frac{\mu}{2}(\alpha^k)^T M^k(\alpha^k) - \mu\boldsymbol{e}^T\alpha^k = F(P^k, \alpha^k), \tag{3.20}$$

where $\mu > 0$ is the hyperparameter introduced in our model.

The optimization problem in (3.1) is a minimization problem and the value of the objective function $F(P^k, \alpha^k)$ in the kth iteration is less than the value of $F(P^{k-1}, \alpha^{k-1})$ in the $(k-1)$st iteration. The following inequality holds for any k

$$F(P^k, \alpha^k) \leq F(P^k, \alpha^{k-1}) \leq F(P^{k-1}, \alpha^{k-1}). \tag{3.21}$$

Inequality (3.21) holds because by fixing α (P) in each iteration and minimizing the problem in (3.1) over P (α) in the next iteration, the value of the objective function decreases.

Inequality (3.20) shows that the sequence $\{F(P^k, \alpha^k)\}_{k=1}^{\infty}$ is bounded below and inequality (3.21) shows that this sequence is decreasing. So the sequence $\{F(P^k, \alpha^k)\}_{k=1}^{\infty}$ is decreasing and bounded below, thus by the **Monotone Convergence** theorem it is convergent.

□

3.3 Multi-class classification

This section is devoted to explaining the strategy that is used in our model to classify a data set with more than two classes. For example, hand-written digit recognition contains 10 different classes: digits 0 to 9. This type of problem is called a "*multi-class classification*" problem. Most classification methods such as SVM are designed for *binary-class* classification problems and are not able to solve these problems directly.

To address this issue, the data set with more than two classes is split into multiple data sets with only two classes and then each binary classification model is trained separately. The most common methods to do multi-class classification are **One-vs-One** and **One-vs-All**. In the One-vs-One method, a multi-class classification problem is split into several binary problems and each problem is solved separately. On the other hand, in the One-vs-All method, a binary model is constructed so that each model is trained with one class as positive, and the rest of the classes as negative. Only the One-vs-One method is discussed in this section. In this method, $q(q-1)/2$ classifiers are constructed, where q is the number of classes, and each one is trained on data with two classes. The One-vs-One method is utilized in our model to do multi-class classification.

For simplicity, consider a data set with 3 different classes. Using the One-vs-One method, this multi-class problem is split into 3 binary classification problems: class 1 vs class 2, class 1 vs class 3, and class 2 vs class 3. For training data for the ith and the jth classes, the following optimization problem needs to be solved:

$$\begin{aligned} \min_{w^{ij},b^{ij},\epsilon^{ij}} \quad & \frac{1}{2}\|w^{ij}\|_2^2 + C\sum_{t=1}^{n}\epsilon_t^{ij} \\ \text{s.t.} \quad & (w^{ij})^T P^T x_t + b^{ij} \geq 1 - \epsilon_t^{ij} \quad \text{, if } x_t \text{ in the } i\text{th class}, \\ & (w^{ij})^T P^T x_t + b^{ij} \geq -1 + \epsilon_t^{ij} \quad \text{, if } x_t \text{ in the } j\text{th class}, \\ & \epsilon_t^{ij} \geq 0. \end{aligned} \tag{3.22}$$

The dual form of the problem (3.22) which is a convex quadratic problem can be solved by MATLAB built-in function quadprog. Once the dual variable α is found, w^{ij} and b^{ij} are calculated by (3.6) and (3.7), respectively. Having w^{ij} and b^{ij}, the classifier $(w^{ij})^T P^T x + b^{ij} = 0$ can be easily obtained.

For a three-class data set, three classifiers (decision functions), denoted by DF^{12}, DF^{13}, and DF^{23} are calculated. In order to classify a new data point $\boldsymbol{x}$, a "*voting*" strategy is used. Using this strategy, the new data point is put into three decision functions. Class i will get a vote if the binary problem of classes i and j indicates the new data point $\boldsymbol{x}$ should belong to class i. In the end, the new data point belongs to a class with the maximum number of votes. In our proposed model, this strategy is used to classify multi-class data sets.

3.4 Numerical experiments

3.4.1 Experimental settings

In this subsection, the performance of various dimensionality reduction methods is evaluated on two toy data sets. These data sets are available from the LIBSVM website[1]. We use "Heart" and "Iris" data sets as toy examples for classification. A brief description of these data sets is given in Table 3.1.

Table 3.1: Description of data sets used in the experiments.

DATA SET	# OBSERVATIONS	# FEATURES	# CLASSES
Heart	270	13	2
Iris	150	4	3

The experimental results are divided into two parts, Part I and Part II. The first part shows the experimental settings and numerical results of the comparison of several dimensionality reduction methods, including PCA, LDA, SPCA, and our proposed model whose algorithm is summarized in Algorithm 3.1. In this part, we look at our model just as a dimensionality reduction method and do not consider its

[1]https://www.csie.ntu.edu.tw/∼cjlin/libsvm/

ability to do classification. The performance of our model as a whole package which does the dimensionality reduction and classification is shown in the second part.

In our experiments, first we conduct a comparison among PCA, LDA, SPCA, and our proposed model whose algorithm is summarized in Algorithm 3.1. We have performed 40 random splits of the data, taking different portions for training and testing sets $(20\%/80\%, 40\%/60\%, 50\%/50\%, 60\%/40\%, 80\%/20\%)$. First, the transformation matrix is computed by these four dimensionality reduction methods being applied to the training set. In the case of SPCA, linear kernel function, $\kappa(x_i, x_j) = {x_i}^T x_j$ is applied to the labels. In our model, we use PCA on the training data to compute the initial projection matrix P. For an initial P obtained by PCA, LIBSVM [46,97] and MATLAB built-in function quadprog are separately employed to solve the α-subproblem (3.2) for an α vector. This vector is used to find the solution of the P-subproblem (3.5) for a projection matrix P. Parameter μ is selected over the range $\{10^{-3}, 10^{-2}, ..., 10^2, 10^3\}$ and the tolerance is set to 10^{-10}. After computing the transformation matrix, the dimension of the testing data is reduced using matrix P and then SVM or kNN ($k = 1, 2, 3$) are used to do classification of the reduced dimension data. We calculate the following quantity to evaluate the performance of Algorithm 3.1:

$$\text{Classification Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \times 100\%.$$

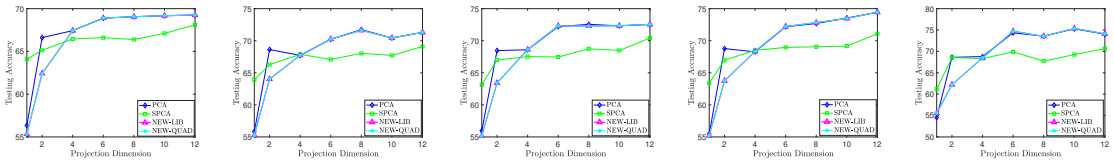
In all methods, while using SVM, different values for the hyperparameter C over the range $\{10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1\}$ and linear kernel are used for the classification task. All experiments are repeated 40 times.

3.4.2 Numerical results: Part I

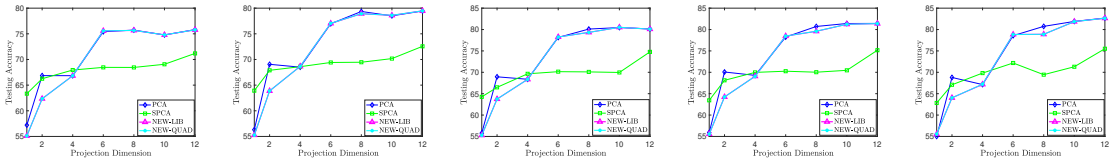
3.4.2.1 Numerical results for "Heart" data set

"Heart" data set, taken from the LIBSVM website [46], is considered as the first toy example to evaluate and compare the performance of dimensionality reduction methods mentioned in the previous subsection. We conduct dimensionality reduction on this data set by PCA, LDA, SPCA, and our proposed model whose convex quadratic problem is solved by LIBSVM and quadprog function. Since "Heart" data set has 13 features, it is projected into low-dimensional spaces with dimension $d = \{1, 2, 4, 6, 8, 10, 12\}$. Due to some limitations the LDA method has, in the case where LDA is used as the dimensionality reduction method, data points in "Heart" data set are projected into a 1-dimensional space. In all figures, the PCA graph represents the classification accuracy of testing data whose dimension is reduced by PCA and is classified by SVM or kNN. SPCA, NEW-LIB, and NEW-QUAD graphs also show the classification accuracy by SVM or kNN of the testing set whose dimension is reduced by SPCA, our proposed model whose convex quadratic optimization problem is solved by LIBSVM, and our proposed model whose convex quadratic optimization problem is solved by quadprog function, respectively.

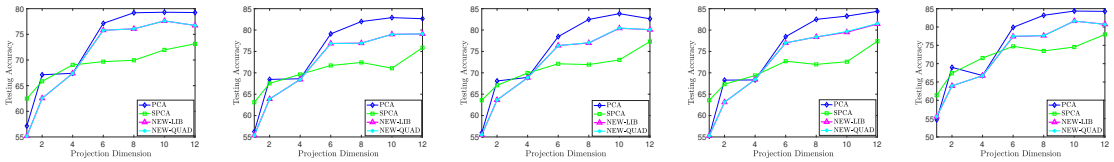
Figure 3.1 shows the classification accuracy on "Heart" data set by SVM as the classification method for different values of hyperparameter C and different portions of training and testing. The results show that our model works well and in most cases, it has the same as or better performance than other methods.



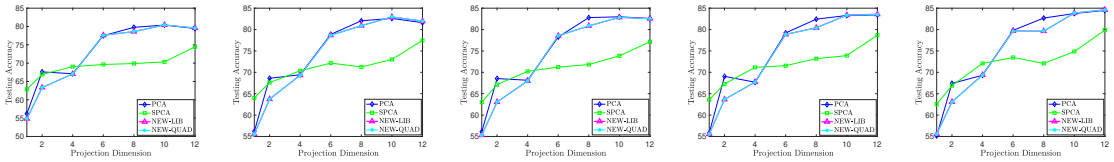
(a) 20% Training (b) 40% Training (c) 50% Training (d) 60% Training (e) 80% Training



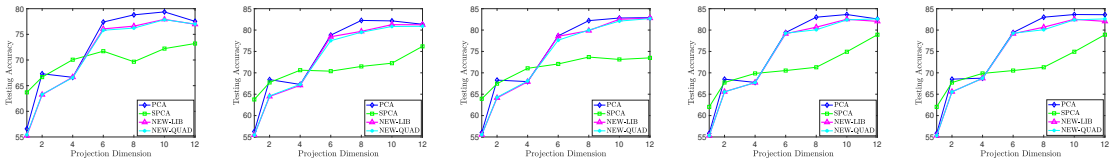
(f) 20% Training (g) 40% Training (h) 50% Training (i) 60% Training (j) 80% Training



(k) 20% Training (l) 40% Training (m) 50% Training (n) 60% Training (o) 80% Training



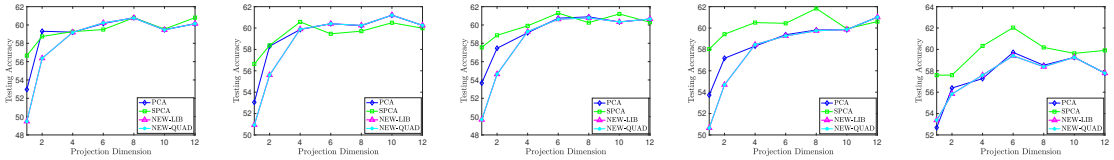
(p) 20% Training (q) 40% Training (r) 50% Training (s) 60% Training (t) 80% Training



(u) 20% Training (v) 40% Training (w) 50% Training (x) 60% Training (y) 80% Training

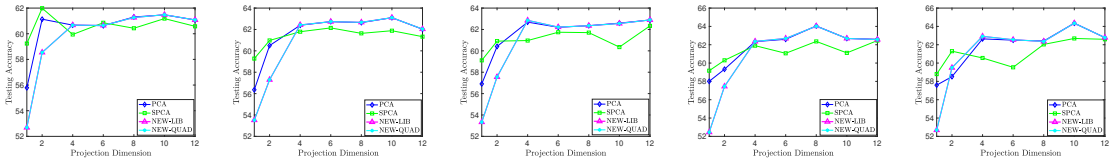
Figure 3.1: Classification accuracy on "Heart" data set by SVM as the classification method with $C = 10^{-3}$ for the first row, $C = 10^{-2}$ for the second row, $C = 10^{-1}$ for the third row, $C = 10^{0}$ for the fourth row, and $C = 10^{1}$ for the fifth row.

The classification accuracy on "Heart" data set by kNN as the classification method for different values of k is also shown in Figures 3.2, 3.3, and 3.4. Similar to the previous case, the results show the good performance of our model.



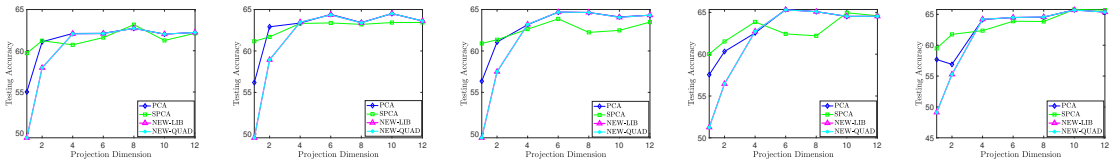
(a) %20 Training (b) %40 Training (c) %50 Training (d) %60 Training (e) %80 Training

Figure 3.2: Classification accuracy on "Heart" data set by kNN with $k = 1$.



(a) 20% Training (b) 40% Training (c) 50% Training (d) 60% Training (e) 80% Training

Figure 3.3: Classification accuracy on "Heart" data set by kNN with $k = 2$.



(a) 20% Training (b) 40% Training (c) 50% Training (d) 60% Training (e) 80% Training

Figure 3.4: Classification accuracy on "Heart" data set by kNN with $k = 3$.

Tables 3.2 and 3.3 contain the classification accuracy on "Heart" data set by SVM or kNN after reducing the dimension of the data by LDA and our proposed model. In Table 3.2, the LDA+SVM row represents the classification accuracy by SVM after applying LDA to project the data into a 1-dimensional space and the NEW+SVM row shows the classification accuracy by SVM after applying our model to project the data into a 1-dimensional space. In Table 3.3, the LDA+kNN row represents the classification accuracy by kNN after applying LDA to the data as the

dimensionality reduction method and the NEW+kNN row shows the classification accuracy by kNN after applying our model to the data as the dimensionality reduction method. The best results are highlighted.

Table 3.2: Classification accuracy (in %) by SVM after applying LDA and our proposed model to "Heart" data set to reduce its dimension.

		mean ± std				
Training Ratio	Model	$C = 10^{-3}$	$C = 10^{-2}$	$C = 10^{-1}$	$C = 10^{0}$	$C = 10^{1}$
20%	**LDA+SVM**	48.44 ± 3.65	51.92 ± 8.44	52.20 ± 10.97	52.77 ± 11.85	52.85 ± 10.35
	NEW+SVM	55.33 ± 0.93	55.11 ± 1.35	55.18 ± 1.05	54.83 ± 2.67	55.41 ± 0.55
40%	**LDA+SVM**	49.70 ± 6.63	48.92 ± 8.58	48.75 ± 9.86	49.47 ± 9.42	49.22 ± 9.43
	NEW+SVM	55.47 ± 0.99	55.26 ± 1.34	55.38 ± 0.64	55.52 ± 0.13	55.46 ± 0.45
50%	**LDA+SVM**	51.64 ± 7.39	48.97 ± 8.88	48.39 ± 9.18	49.73 ± 8.40	48.41 ± 8.14
	NEW+SVM	55.76 ± 1.15	55.33 ± 1.15	55.50 ± 0.35	55.29 ± 1.09	55.51 ± 0.23
60%	**LDA+SVM**	52.87 ± 7.46	47.44 ± 6.66	47.20 ± 6.59	46.67 ± 5.73	47.27 ± 4.70
	NEW+SVM	55.80 ± 1.49	55.55 ± 0.94	55.69 ± 1.23	55.70 ± 1.31	55.62 ± 1.04
80%	**LDA+SVM**	**53.75 ± 7.35**	45.15 ± 1.99	45.85 ± 1.27	44.89 ± 1.62	47.41 ± 4.98
	NEW+SVM	55.91 ± 1.16	55.86 ± 1.24	**55.98 ± 1.42**	55.83 ± 1.85	55.81 ± 0.68

Table 3.3: Classification accuracy (in %) by kNN after applying LDA and our proposed model to "Heart" data set to reduce its dimension.

		mean ± std		
Training Ratio	Model	$k=1$	$k=2$	$k=3$
20%	**LDA+*k*NN**	52.59 ± 10.96	**53.77 ± 11.93**	52.98 ± 12.33
	NEW+*k*NN	49.47 ± 2.69	52.68 ± 2.33	49.44 ± 2.11
40%	**LDA+*k*NN**	50.19 ± 9.51	53.31 ± 11.34	48.86± 9.06
	NEW+*k*NN	50.92 ± 2.10	**53.81 ± 2.36**	49.52 ± 1.58
50%	**LDA+*k*NN**	48.83 ± 7.59	50.87 ± 9.47	48.28 ± 7.94
	NEW+*k*NN	49. 85 ± 2.89	53.33 ± 1.83	49.58 ± 1.35
60%	**LDA+*k*NN**	47.76 ± 4.85	50.41 ± 8.19	47.10 ± 5.84
	NEW+*k*NN	50.64 ± 1.46	52.43 ± 1.21	51.25 ± 1.64
80%	**LDA+*k*NN**	46.63 ± 3.12	48.46 ± 4.48	45.55 ± 2.64
	NEW+*k*NN	53.37 ± 1.04	52.68 ± 1.34	49.12 ± 1.84

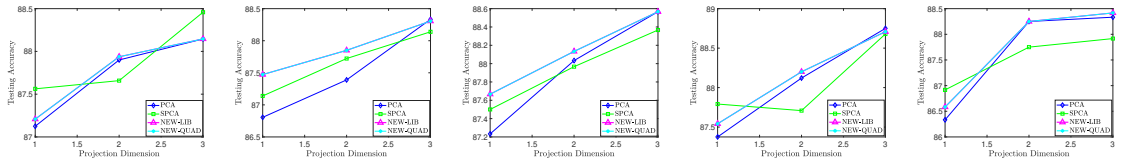
The results show that the performance of our model is by far better than the performance of LDA as the dimensionality reduction method prior to the classification.

3.4.2.2 Numerical results for "Iris" data set

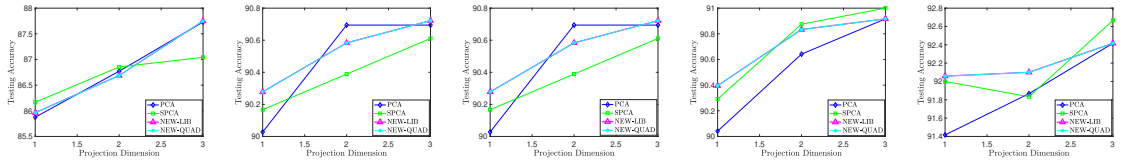
"Iris" data set, taken from the LIBSVM website [46], is another toy example considered in our experiments to evaluate and compare the performance of dimensionality reduction methods mentioned in the previous subsection. We conduct dimensionality reduction on this data set by PCA, LDA, SPCA, and our proposed model whose convex quadratic problem is solved by LIBSVM and quadprog function, respectively. We project the data into low-dimensional spaces with dimension $d = \{1, 2, 3\}$. Due to the limitations of the LDA method, in the case where LDA is used as the dimensionality reduction method, data points in "Iris" data set are projected into a 2-dimensional

space. In all figures, the PCA graph represents the classification accuracy of testing data whose dimension is reduced by PCA and is classified by SVM or kNN. SPCA, NEW-LIB, and NEW-QUAD graphs also show the classification accuracy by SVM or kNN of the testing set whose dimension is reduced by SPCA, our proposed model whose convex quadratic problem is solved by LIBSVM, and our proposed model whose convex quadratic problem is solved by quadprog function, respectively.

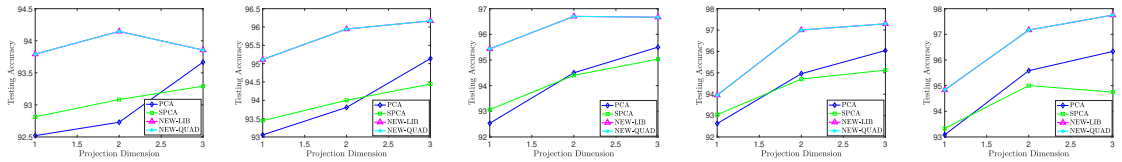
Figures 3.5-3.8 show the classification accuracy on "Iris" data set by SVM and kNN as the classification methods for different values of C and k, and different portions of training and testing, respectively. The results show that our model works well and in most cases, it has the same as or better performance than other methods.



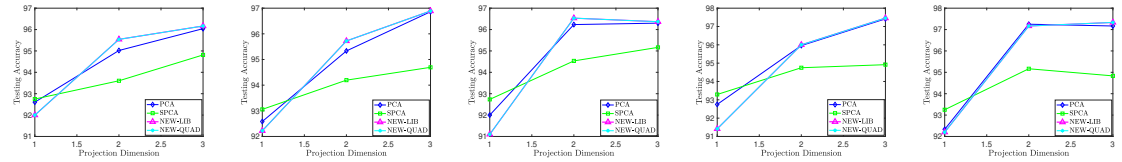
(a) 20% Training (b) 40% Training (c) 50% Training (d) 60% Training (e) 80% Training



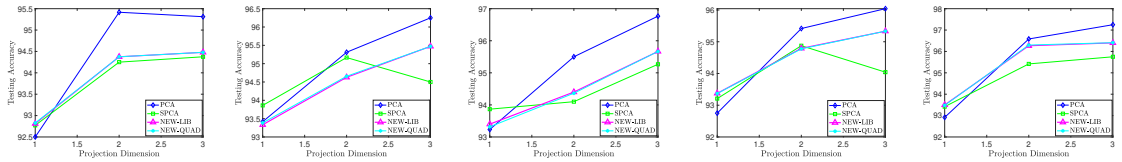
(f) 20% Training (g) 40% Training (h) 50% Training (i) 60% Training (j) 80% Training



(k) 20% Training (l) 40% Training (m) 50% Training (n) 60% Training (o) 80% Training

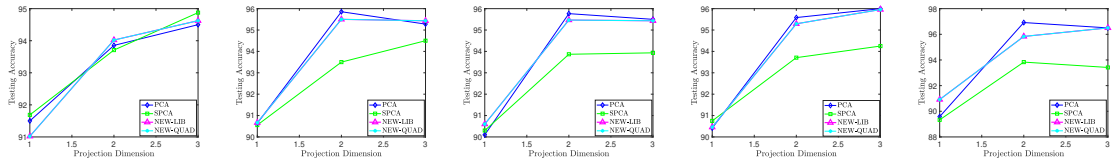


(p) 20% Training (q) 40% Training (r) 50% Training (s) 60% Training (t) 80% Training



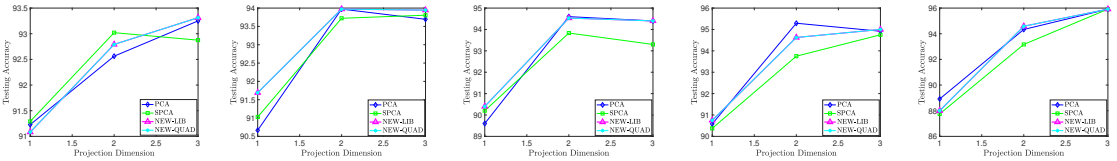
(u) 20% Training (v) 40% Training (w) 50% Training (x) 60% Training (y) 80% Training

Figure 3.5: Classification accuracy on "Iris" data set by SVM as the classification method with $C = 10^{-3}$ for the first row, $C = 10^{-2}$ for the second row, $C = 10^{-1}$ for the third row, $C = 10^{0}$ for the fourth row, and $C = 10^{1}$ for the fifth row.



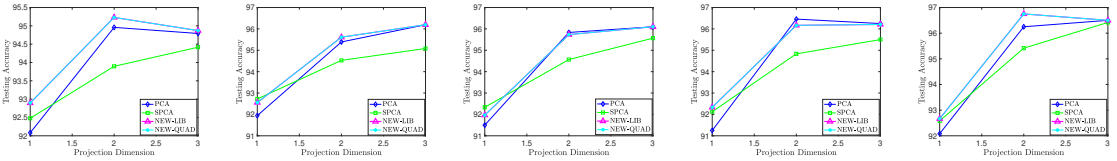
(a) 20% Training (b) 40% Training (c) 50% Training (d) 60% Training (e) 80% Training

Figure 3.6: Classification accuracy on "Iris" data set by kNN with $k = 1$.



(a) 20% Training (b) 40% Training (c) 50% Training (d) 60% Training (e) 80% Training

Figure 3.7: Classification accuracy on "Iris" data set by kNN with $k = 2$.



(a) 20% Training (b) 40% Training (c) 50% Training (d) 60% Training (e) 80% Training

Figure 3.8: Classification accuracy on "Iris" data set by kNN with $k = 3$.

Tables 3.4 and 3.5 contain the classification accuracy on "Iris" data set by SVM or kNN after reducing the dimension of the data by LDA and our proposed model. In Table 3.4, the LDA+SVM row represents the classification accuracy by SVM after applying LDA to project the data into a 2-dimensional space and the NEW+SVM row shows the classification accuracy by SVM after applying our model to project the data into a 2-dimensional space. In Table 3.5, the LDA+kNN row represents the classification accuracy by kNN after applying LDA to project the data into a

2-dimensional space and the NEW+kNN row shows the classification accuracy by kNN after applying our model to the data as the dimensionality reduction method and projecting the data into a 2-dimensional space. The best results are highlighted. It can be noted that our model outperforms LDA as the dimensionality reduction method prior to the classification.

Table 3.4: Classification accuracy (in %) by SVM after applying LDA and our proposed model to "Iris" data set to reduce its dimension.

		mean ± std				
Training Ratio	Model	$C = 10^{-3}$	$C = 10^{-2}$	$C = 10^{-1}$	$C = 10^{0}$	$C = 10^{1}$
20%	**LDA+SVM**	74.63 ± 16.44	69.37 ± 18.96	69.11 ± 19.02	66.18 ± 18.66	66.73 ± 19.01
	NEW+SVM	87.93 ± 1.09	86.68 ± 1.29	94.14 ± 1.56	95.54 ± 1.84	94.37 ± 2.22
40%	**LDA+SVM**	**79.73 ± 11.70**	73.69 ± 11.68	70.55 ± 16.28	67.32 ± 18.01	71.68 ± 16.84
	NEW+SVM	87.85 ± 1.50	90.58 ± 2.09	95.94 ± 1.95	95.72 ± 1.63	94.62 ± 1.44
50%	**LDA+SVM**	78.68 ± 11.08	72.06 ± 10.12	70.44 ± 14.21	69.32 ± 15.40	70.93 ± 14.00
	NEW+SVM	88.13 ± 1.23	90.40 ± 2.02	96.70 ± 1.62	96.53 ± 1.78	94.40 ± 1.76
60%	**LDA+SVM**	75.94 ± 8.37	71.70 ± 9.91	68.18 ± 14.05	66.58 ± 15.44	69.11 ± 14.66
	NEW+SVM	88.20 ± 1.96	90.83 ± 2.11	97.00 ± 1.96	96.00 ± 1.80	94.79 ± 1.43
80%	**LDA+SVM**	74.27 ± 5.96	71.55 ± 5.42	68.16 ± 13.70	63.08 ± 12.07	64.61 ± 11.17
	NEW+SVM	88.25 ± 1.33	92.10 ± 1.28	97.16 ± 1.54	**97.19 ± 1.48**	96.26 ± 1.91

Table 3.5: Classification accuracy (in %) by kNN after applying LDA and our proposed model to "Iris" data set to reduce the dimension.

		mean ± std		
Training Ratio	Model	$k = 1$	$k = 2$	$k = 3$
20%	**LDA+*k*NN**	72.09 ± 13.73	73.34 ± 16.12	74.55 ± 13.37
	NEW+*k*NN	94.02 ± 1.84	92.79 ± 2.52	95.22 ± 2.24
40%	**LDA+*k*NN**	71.19 ± 7.73	74.92 ± 10.34	71.65 ± 8.63
	NEW+*k*NN	95.51 ± 1.81	93.97 ± 2.56	95.61± 1.81
50%	**LDA+*k*NN**	72.56 ± 7.87	**74.64 ± 8.85**	70.66 ± 7.52
	NEW+*k*NN	95.46 ± 1.63	94.53 ± 2.43	95.73 ± 2.16
60%	**LDA+*k*NN**	71.90 ± 6.20	73.61 ± 6.86	71.77 ± 6.38
	NEW+*k*NN	95.29 ± 1.24	94.62 ± 2.78	96.15 ± 1.69
80%	**LDA+*k*NN**	70.88 ± 4.06	71.50 ± 4.24	70.08 ± 4.25
	NEW+*k*NN	95.83 ± 2.08	94.58 ± 2.51	**96.75 ± 1.24**

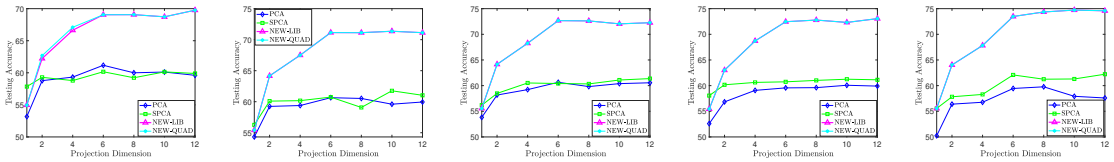
3.4.3 Numerical results: Part II

In our experiments, we also compare the classification accuracy by our model and the one by SVM and kNN after reducing the dimension of the data by PCA, LDA, and SPCA. Experimental settings are similar to the ones we have in the previous subsections. To compare the classification accuracy, first we apply PCA and SPCA to the training set and find the projection matrix. This projection matrix is used to reduce the dimension of the testing set. After reducing the dimension, SVM and kNN are employed to obtain the classification accuracy. On the other hand, in our model, we use PCA on the training data to compute the initial projection matrix P, and we alternatingly solve problems (3.2) for an α and (3.5) for a transformation matrix. This transformation matrix is used to reduce the dimension of the testing data. Our model also learns a classifier while calculating the projection matrix. After

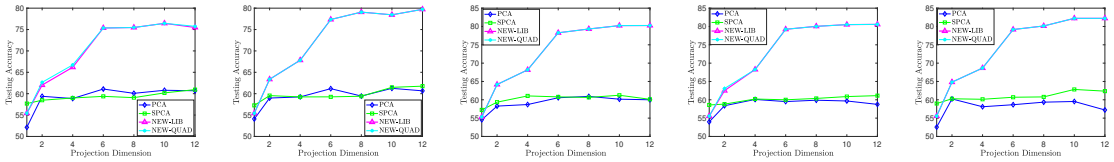
reducing the dimension of the testing data, the classifier hyperplane found by our model is applied to the testing data to obtain the classification accuracy. In our model, the label/class of each unseen data point in the reduced dimension testing data is obtained by the **decision function** in (3.8).

3.4.3.1 Numerical results for "Heart" data set

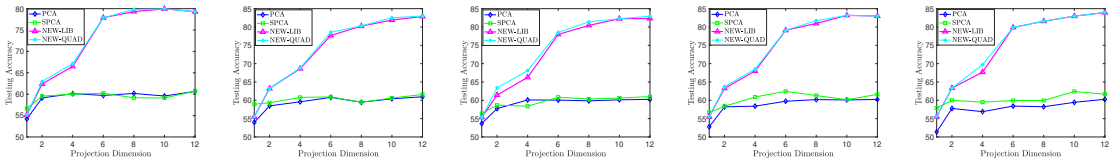
The classification accuracy by the decision function for different values of hyperparameter C is presented in Figure 3.9. In this figure, the decision function in (3.8) is only used to find the classification accuracy of the testing data whose dimension is reduced by our proposed model solved by LIBSVM or quadprog function. For "Heart" data set, 1NN is used to find the classification accuracy of the testing data for which the dimension is reduced by PCA and SPCA. Figure 3.9 shows that our model outperforms PCA and SPCA and achieves much better results than these two methods.



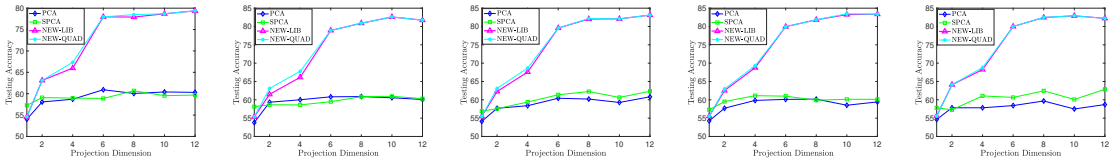
(a) 20% Training (b) 40% Training (c) 50% Training (d) 60% Training (e) 80% Training



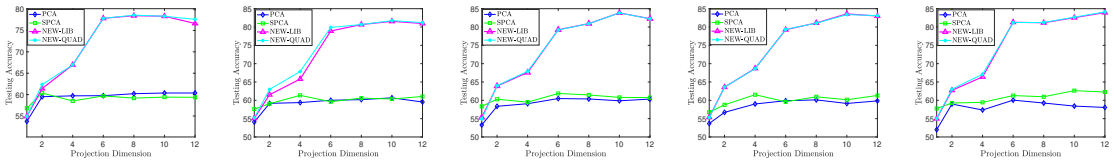
(f) 20% Training (g) 40% Training (h) 50% Training (i) 60% Training (j) 80% Training



(k) 20% Training (l) 40% Training (m) 50% Training (n) 60% Training (o) 80% Training



(p) 20% Training (q) 40% Training (r) 50% Training (s) 60% Training (t) 80% Training



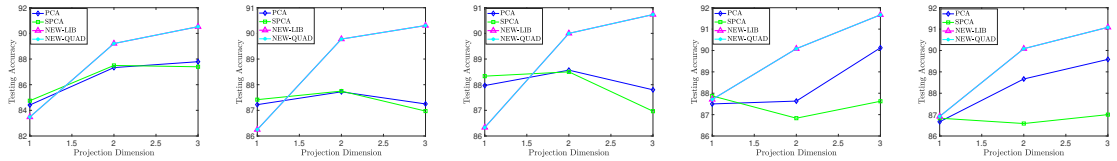
(u) 20% Training (v) 40% Training (w) 50% Training (x) 60% Training (y) 80% Training

Figure 3.9: Classification accuracy on "Heart" data set by the decision function in our proposed model as the classification method with $C = 10^{-3}$ for the first row, $C = 10^{-2}$ for the second row, $C = 10^{-1}$ for the third row, $C = 10^{0}$ for the fourth row, and $C = 10^{1}$ for the fifth row. kNN with $k = 1$ is applied on the testing data whose dimension is reduced by PCA and SPCA.

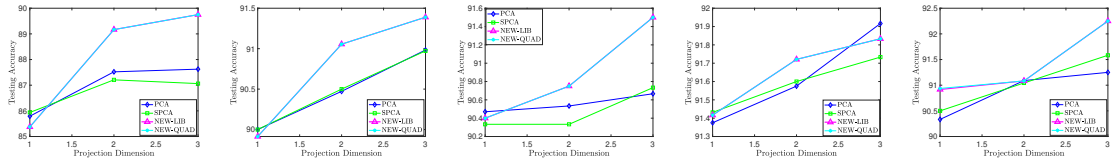
3.4.3.2 Numerical results for "Iris" data set

The classification accuracy by decision function applied on "Iris" data set is in Figure 3.10. While using the decision function for classifying the "Iris" data set by

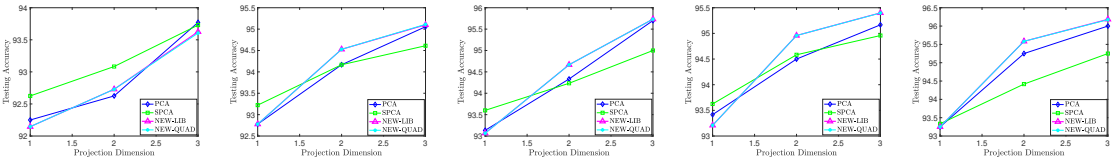
our proposed model, SVM is used to find the classification accuracy of the data set for which the dimension is reduced by PCA and SPCA. From Figure 3.10, it can be seen that our model works well and often it outperforms other methods. This figure also indicates the ability of our model to solve multi-class classification problems.



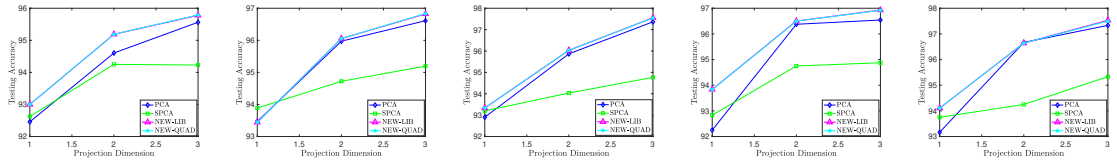
(a) 20% Training (b) 40% Training (c) 50% Training (d) 60% Training (e) 80% Training



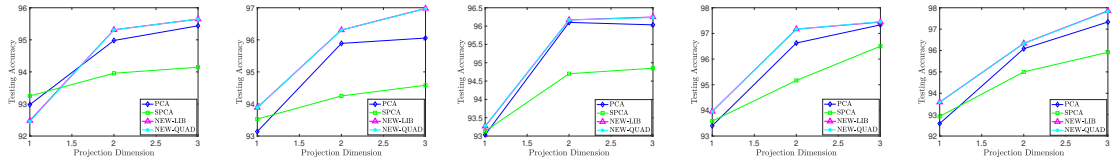
(f) 20% Training (g) 40% Training (h) 50% Training (i) 60% Training (j) 80% Training



(k) 20% Training (l) 40% Training (m) 50% Training (n) 60% Training (o) 80% Training



(p) 20% Training (q) 40% Training (r) 50% Training (s) 60% Training (t) 80% Training



(u) 20% Training (v) 40% Training (w) 50% Training (x) 60% Training (y) 80% Training

Figure 3.10: Classification accuracy on "Iris" data set by the decision function in our proposed model as the classification method with $C = 10^{-3}$ for the first row, $C = 10^{-2}$ for the second row, $C = 10^{-1}$ for the third row, $C = 10^{0}$ for the fourth row, and $C = 10^{1}$ for the fifth row. SVM is applied on the testing data whose dimension is reduced by PCA and SPCA.

CHAPTER 4

MOVING BEYOND LINEARITY

4.1 Introduction

In many cases, not all high-dimensional data problems are linearly separable and there is a nonlinear relationship between the outcome and the predictors, i.e., the decision boundary between the two classes is nonlinear. We can take image data as an example of high-dimensional data whose pixels are not very informative. In this case, linear classifiers perform poorly. To accommodate a nonlinear boundary between different classes in data sets, the feature space needs to be enlarged using nonlinear mappings. Although the decision boundary in the original feature space is nonlinear, however, it will be linear in the enlarged feature space and this is a nice property of the new feature space; see Figure 4.1.

Suppose the following nonlinear mapping is used to enlarge the feature space:

$$\Phi : x \to \Phi(x) \in \mathcal{H},$$

where $\mathcal{H}$ is a Hilbert space that represents the feature space. Φ embeds vector x into the space $\mathcal{H}$.

In this chapter, we extend our model to a nonlinear form to be applied to data sets with nonlinear and complex structures. We assume that the nonlinear mapping Φ is explicitly known. The following problem is the optimization problem of our nonlinear model:

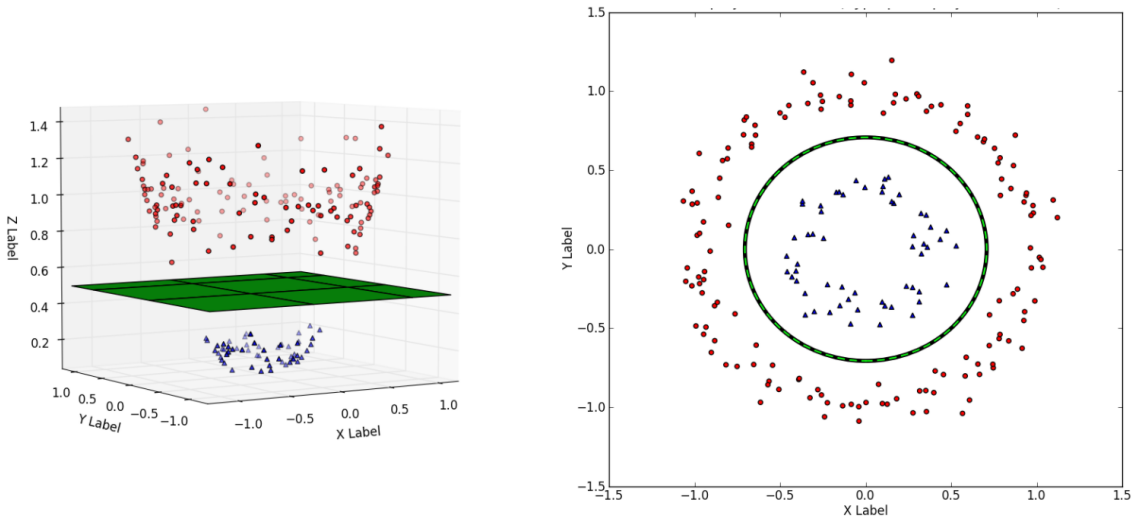


Figure 4.1: The decision boundary is nonlinear before transforming data to a higher-dimensional space (Right). After enlarging the feature space, the decision boundary becomes linear (Left) [117].

$$\begin{aligned} \min_{P,\alpha} \quad & F(P,\alpha) = \|\Phi(X) - PP^T\Phi(X)\|_F^2 + \mu(\frac{1}{2}\alpha^T A^T PP^T A\alpha - \boldsymbol{e}^T\alpha) \qquad (4.1) \\ \text{s.t.} \quad & P^T P = I_d, \\ & \sum_{i=1}^{n} z_i\alpha_i = 0, \\ & 0 \leq \alpha_i \leq C, \qquad i = 1, \cdots, n, \end{aligned}$$

where $\Phi(X) = [\Phi(x_1), \Phi(x_2), ..., \Phi(x_n)] \in \mathbb{R}^{D\times n}$, $P \in \mathbb{R}^{D\times d}$, $Z = [z_1, z_2, ..., z_n] \in \mathbb{R}^{1\times n}$, $\alpha \in \mathbb{R}^n$, and $\boldsymbol{e} \in \mathbb{R}^n$ are the new data matrix, projection matrix, label matrix, the dual variable, and the vector of all ones, respectively, with D ($D \geqslant p$) being the number of features (dimension of the data) in the enlarged feature space, n being the number of observations, and d being the dimension of the low-dimensional space. $A = [z_1\Phi(x_1), \cdots, z_n\Phi(x_n)] \in \mathbb{R}^{D\times n}$ and $C > 0$ and $\mu > 0$ are hyperparameters in our model that need to be tuned. New data matrix $\Phi(X)$ and label matrix Z are known. Φ is a nonlinear mapping used for enlarging the feature space. The goal of our model

is to find the projection matrix P being used in dimensionality reduction and to find the classifier hyperplane $w^T P^T \Phi(x) + b = 0$ being used for data classification. With the help of the dual variable α, w and b which are the normal vector and y-intercept of the separating hyperplane, respectively, can be obtained.

Problem (4.1) is non-convex, but it can be solved in an alternating way as follows:

- α-subproblem (fixing P)

$$\begin{aligned} \min_{\alpha \in \mathbb{R}^n} \quad & f(\alpha) = \frac{1}{2}\alpha^T A^T P P^T A \alpha - \boldsymbol{e}^T \alpha \\ \text{s.t.} \quad & \sum_{i=1}^{n} z_i \alpha_i = 0, \\ & 0 \le \alpha_i \le C, \qquad i = 1, \cdots, n; \end{aligned} \tag{4.2}$$

- P-subproblem (fixing α)

$$\begin{aligned} \min_{P \in \mathbb{R}^{D \times d}} \quad & g(P) = \|\Phi(X) - P P^T \Phi(X)\|_F^2 + \frac{\mu}{2}\alpha^T A^T P P^T A \alpha \\ \text{s.t.} \quad & P^T P = I_d. \end{aligned} \tag{4.3}$$

Problem (4.3) can be converted into a trace optimization problem. In fact,

$$\begin{aligned} g(P) :&= \|\Phi(X) - P P^T \Phi(X)\|_F^2 + \frac{\mu}{2}\alpha^T A^T P P^T A \alpha \\ &= \|\Phi(X)\|_F^2 - \|P^T \Phi(X)\|_F^2 + \frac{\mu}{2}\|P^T A \alpha\|_2^2 \\ &= \mathbf{Tr}(\Phi(X)^T \Phi(X)) - \mathbf{Tr}(P^T \Phi(X) \Phi(X)^T P) + \frac{\mu}{2}\mathbf{Tr}(P^T A \alpha \alpha^T A^T P) \\ &= \mathbf{Tr}(\Phi(X)^T \Phi(X)) + \mathbf{Tr}(P^T T P), \end{aligned} \tag{4.4}$$

where $T = \frac{\mu}{2} A \alpha \alpha^T A^T - \Phi(X)\Phi(X)^T$. Hence, the P-subproblem (4.3) is equivalent to:

- P-subproblem (fixing α)

$$\begin{aligned} \min_{P \in \mathbb{R}^{D \times d}} \quad & g(P) = \mathbf{Tr}(P^T T P) \\ \text{s.t.} \quad & P^T P = I_d. \end{aligned} \tag{4.5}$$

4.1.1 Solving subproblems (4.2) and (4.5)

Similar to the linear form of our model, the optimization problem (4.2) is a convex quadratic optimization problem that can be solved by a large scale quadratic programming solver such as MATLAB built-in function quadprog or LIBSVM package [46, 97]. Problem (4.5) is equivalent to finding the smallest d eigenvalues of matrix T [112]. We can solve problems (4.2) for α and (4.5) for P, alternatingly until convergence.

The outputs of our alternating algorithm are the projection matrix P and the dual variable α. P is directly utilized to reduce the dimension of the data and α helps to find the classifier hyperplane. As discussed earlier, with the help of the KKT optimality conditions of SVM optimization problem, after solving (4.2) and obtaining α, the normal vector, w, and the y-intercept, b, of the classifier hyperplane can be calculated by:

$$w = \sum_{i=1}^{n} \alpha_i z_i P^T \Phi(x_i) \tag{4.6}$$

and

$$b = -\frac{\sum_{i:0<\alpha_i<C} z_i \nabla f(\alpha)_i}{|\{i : 0 < \alpha_i < C\}|}, \tag{4.7}$$

where $\nabla f(\alpha) = (A^T P P^T A)\alpha - \boldsymbol{e}$ is the gradient of the objective function in (4.2).

Once all variables are found, the label of the new and unseen data $\boldsymbol{x}$ (testing data) can be predicted by the following decision function:

$$DF(\boldsymbol{x}) = \text{sgn}(w^T P^T \Phi(\boldsymbol{x}) + b) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i z_i \Phi(x_i)^T P P^T \Phi(\boldsymbol{x}) + b\right), \tag{4.8}$$

where sgn is the sign function.

Algorithm 4.1 summarizes the procedure of our nonlinear supervised model.

Algorithm 4.1 Nonlinear Supervised PCA-SVM

Input: $X, P^0, Z, \mu > 0, C > 0, d$, tolerance, and $k = 1$;

Output: $P \in \mathbb{R}^{D\times d}, \alpha \in \mathbb{R}^n$.

construct matrix $A = [z_1\Phi(x_1), \cdots, z_n\Phi(x_n)]$;

while $|F(P^{k+1}, \alpha^{k+1}) - F(P^k, \alpha^k)| \geq$ tolerance **do**

  for given P^k, solve α-subproblem (4.2) for α^k;

  $T^k = \frac{\mu}{2} A(\alpha^k)(\alpha^k)^T A^T - \Phi(X)\Phi(X)^T$;

  find eigenvalues and eigenvectors of T^k;

  sort eigenvalues in the ascending order;

  let P^k be composed of the eigenvectors corresponding to the d smallest eigenvalues of T^k;

  $k \leftarrow k + 1$;

end while

4.2 Numerical experiments

4.2.1 Experimental settings

In this subsection, the performance of various dimensionality reduction methods is evaluated on 5 data sets. These data sets are available from the LIBSVM website[1]. A brief description of the data sets is given in Table 4.1.

Table 4.1: Description of data sets used in the experiments

DATA SET	# OBSERVATIONS	# FEATURES	# CLASSES
sonar	208	60	2
german.numer	1000	24	2
wine	178	13	3
svmguide2	391	20	3
dna	2000	180	3

The experimental results are divided into two parts. The first part shows the experimental settings and results of the comparison of several dimensionality reduction methods, including PCA, LDA, SPCA, and our proposed model whose algorithm is summarized in Algorithm 4.1. In this part, we look at our model just as a dimensionality reduction algorithm and do not consider its ability to do classification. The performance of our model as a whole package which does the dimensionality reduction and classification is shown in the second part.

4.2.2 Nonlinear mapping Φ

A polynomial kernel function is:

$$\kappa(x_i, x_j) = (\gamma x_i^T x_j + r)^m = \langle \Phi(x_i), \Phi(x_j) \rangle, \tag{4.9}$$

[1]https://www.csie.ntu.edu.tw/ cjlin/libsvm/

where m is the degree of the polynomial kernel function, and $\gamma > 0$ and r are the parameters for this kernel. Similar to other kernel functions, the polynomial kernel is computed by the inner product of two vectors $\Phi(x_i)$ and $\Phi(x_j)$. Parameter r can be fixed to one [118]. In all experiments, we use the following degree-2 nonlinear polynomial mapping:

$$\Phi(x) = [1, \sqrt{2\gamma}x_1, \cdots, \sqrt{2\gamma}x_p, \gamma x_1^2, \cdots, \gamma x_p^2, \sqrt{2\gamma}x_1x_2, \cdots, \sqrt{2\gamma}x_{p-1}x_p]^T. \quad (4.10)$$

4.2.3 Parameter selection

Our nonlinear model has 5 parameters (C, μ, γ, r, m) that need to be tuned. Since we set r to be one and only use the polynomial with degree 2, the number of parameters is reduced to 3. In all experiments, parameters C and γ are chosen by a 10-fold cross-validation over the range $\{10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2, 10^3\}$. Parameter μ is also chosen over the same range.

4.2.4 Numerical results: Part I

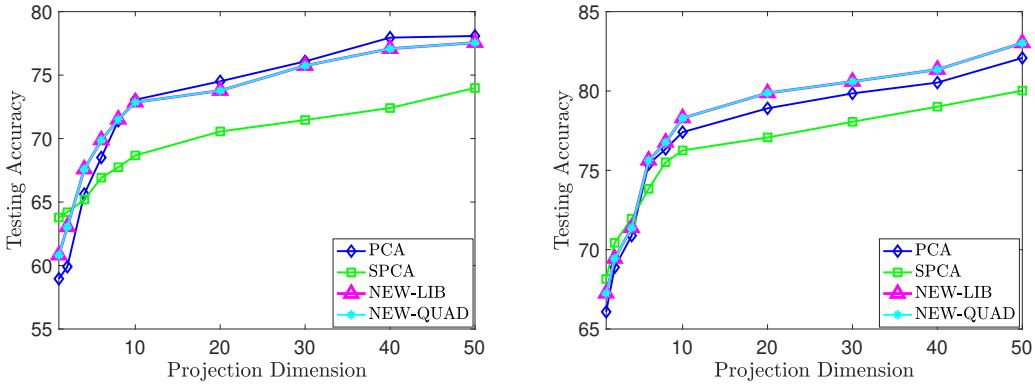
In this subsection, we look at our model just as a dimensionality reduction method and do not consider its ability to do classification. The experimental settings and numerical results of the comparison of several dimensionality reduction methods, including PCA, LDA, SPCA, and our proposed model are also shown in this subsection. In our experiments, we conduct a comparison among PCA, LDA, SPCA, and our proposed model in Algorithm 4.1. We have performed 40 random splits of the data and use a %80/%20 split for training and testing. First, the transformation matrix is computed by these four dimensionality reduction methods being applied to the training set. In the case of SPCA, linear kernel function, $\kappa(x_i, x_j) = {x_i}^T x_j$ is applied to the labels. In our model, we use PCA on the training data to compute the initial projection matrix P. For given P, LIBSVM [46, 97] and MATLAB built-in quadprog

are separately employed to solve the α-subproblem (4.2) for an α vector. This vector is used to find the solution of subproblem (4.5) for a projection matrix P. The tolerance is set to 10^{-10}. After computing the transformation matrix, the dimension of the testing data is reduced using matrix P and and then SVM or 1NN are used to do classification of the reduced dimension data. LDA limitations force to project the data sets with 2 classes into a 1-dimensional space and the data sets with 3 classes into a 2-dimensional space. We calculate the following quantity to evaluate the performance of Algorithm 4.1:

$$\text{Classification Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \times 100\%.$$

In all figures, the PCA graph represents the classification accuracy of testing data whose dimension is reduced by PCA and is classified by SVM or 1NN. SPCA, NEW-LIB, and NEW-QUAD graphs also show the classification accuracy by SVM or 1NN of the testing data whose dimension is reduced by SPCA, our proposed model whose convex quadratic problem is solved by LIBSVM, and our proposed model whose convex quadratic problem is solved by quadprog function, respectively. All experiments are repeated 40 times.

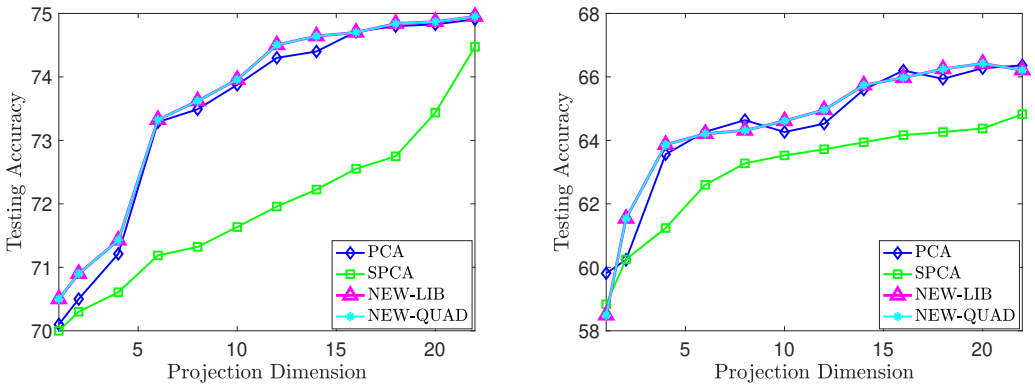
Figure 4.2 shows the classification accuracy on "sonar" data set by SVM and 1NN as the classification methods for the best values of hyperparameters C, μ, and γ. This data set is projected into low-dimensional spaces with dimension $d = \{1, 2, 4, 6, 8, 10, 20, 30, 40, 50\}$. The results show that our model works well and in most cases, it has the same as or better performance than other methods.



(a) SVM (b) 1NN

Figure 4.2: Classification accuracy on "sonar" data set. (a) Classification accuracy by SVM. (b) Classification accuracy by 1NN. The best $C = 1000, \mu = 10$, and $\gamma = 0.1$.

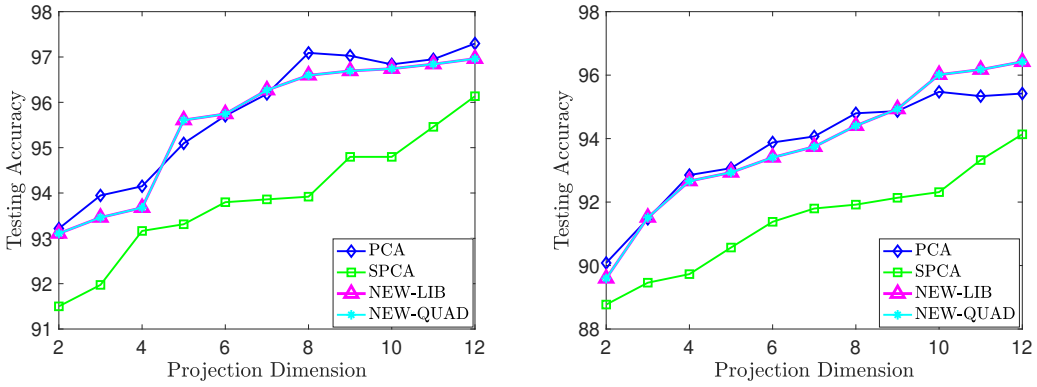
The classification accuracy on "german.numer" data set is shown in Figure 4.3. The classification accuracy is obtained by SVM and 1NN as the classification methods for the best values of hyperparameters C, μ, and γ. This data set is projected into low-dimensional spaces with dimension $d = \{1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22\}$. The results show that how our model outperforms against other models.



(a) SVM (b) 1NN

Figure 4.3: Classification accuracy on "german.numer" data set. (a) Classification accuracy by SVM. (b) Classification accuracy by 1NN. The best $C = 100, \mu = 1$, and $\gamma = 0.001$.

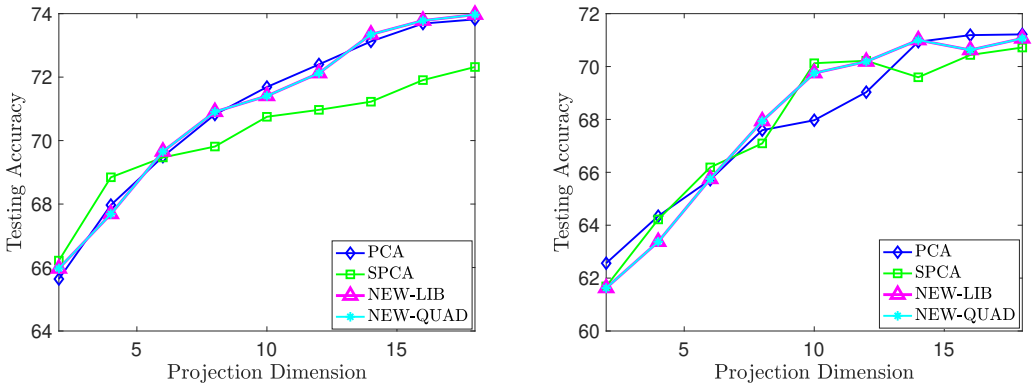
The classification accuracy on "wine" data set is shown in Figure 4.4. The classification accuracy is obtained by SVM and 1NN as the classification methods for the best values of hyperparameters C, μ, and γ. This data set is projected into low-dimensional spaces with dimension $d = \{2, 4, 6, 8, 10, 12\}$. The results show that our model has the same as or better performance than by other models.



(a) SVM (b) 1NN

Figure 4.4: Classification accuracy on "wine" data set. (a) Classification accuracy by SVM. (b) Classification accuracy by 1NN. The best $C = 10, \mu = 10$, and $\gamma = 0.01$.

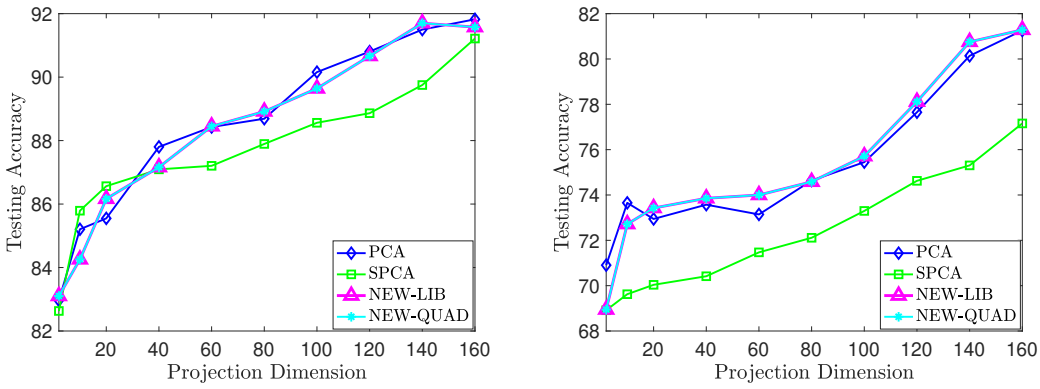
The classification accuracy on "svmguide2" data set is shown in Figure 4.3. The classification accuracy is obtained by SVM and 1NN as the classification methods for the best values of hyperparameters C, μ, and γ. This data set is projected into low-dimensional spaces with dimension $d = \{2, 4, 6, 8, 10, 12, 14, 16, 18\}$. The results show that our model works well and in most cases, it has the same as or better performance than other methods.



(a) SVM (b) 1NN

Figure 4.5: Classification accuracy on "svmguide2" data set. (a) Classification accuracy by SVM. (b) Classification accuracy by 1NN. The best $C = 1, \mu = 0.1$, and $\gamma = 100$.

Figure 4.2 shows the classification accuracy on "dna" data set by SVM and 1NN as the classification methods for the best values of hyperparameters C, μ, and γ. This data set is projected into low-dimensional spaces with dimension $d = \{2, 10, 20, 40, 60, 80, 100, 120, 140, 160\}$. The results show that our model works well and has a similar performance to PCA. It can be easily seen that our model outperforms the SPCA model.



(a) SVM (b) 1NN

Figure 4.6: Classification accuracy on "dna" data set. (a) Classification accuracy by SVM. (b) Classification accuracy by 1NN. The best $C = 1, \mu = 10$, and $\gamma = 0.01$.

Tables 4.2 contains the classification accuracy on 6 data sets by SVM or 1NN after after reducing the dimension of the data by LDA and our proposed model. In this table, the LDA+SVM and LDA+1NN rows represent the classification accuracy by SVM and 1NN, respectively, after applying LDA as the dimensionality reduction method to different data sets. The NEW+SVM and NEW+1NN rows show the classification accuracy obtained by SVM and 1NN, respectively, after applying our model as the dimensionality reduction method to different data sets. The best results are shown in this table. It can be noted that, almost in all cases, our model outperforms LDA and its performance is much better than LDA.

Table 4.2: Classification accuracy (in %) by SVM and 1NN after applying LDA and our proposed model to different data sets with the best parameters C, μ, and γ.

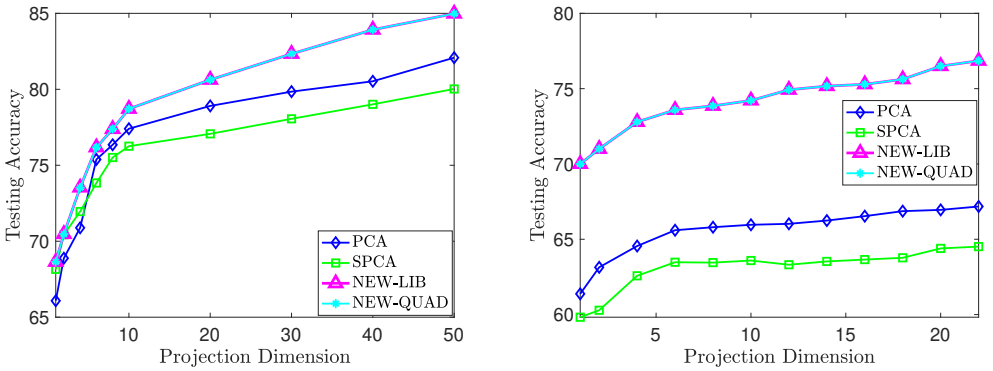
	mean ± std				
Model/Data Set	sonar	german.numer	wine	svmguide2	dna
LDA+SVM	59.08 ± 6.69	69.03 ± 1.28	92.14 ± 3.02	64.84 ± 9.54	80.28 ± 3.25
NEW+SVM	60.81 ± 1.23	70.50 ± 2.26	93.10 ± 2.31	65.96 ± 2.18	83.10 ± 1.66
LDA+1NN	64.77 ± 5.97	61.55 ± 0.96	88.10 ± 2.75	60.14 ± 8.64	68.46 ± 3.62
NEW+1NN	68.64 ± 2.08	58.78 ± 1.84	92.25 ± 1.37	61.62 ± 1.78	68.94 ± 2.09

4.2.5 Numerical results: Part II

The performance of our model in Algorithm 4.1 as a whole package which does the dimensionality reduction and classification is shown in this subsection. In our experiments, we also compare the classification accuracy by our model and the one obtained by 1NN after reducing the dimension of the data by PCA and SPCA. Experimental settings are similar to the ones we have in the previous subsection. To compare the classification accuracy, first we apply PCA and SPCA to the training set and find the projection matrix. This projection matrix is used to reduce the dimension of the testing data. After reducing the dimension, 1NN is employed to obtain the classification accuracy. On the other hand, in our model, we use PCA on the training data to compute the initial projection matrix P and alternatingly solve problems (4.2) for an α vector and (4.5) for a transformation matrix. This transformation matrix is used to reduce the dimension of the testing data. Our model also learns the classifier while calculating the projection matrix. After reducing the dimension of the testing set, the classifier hyperplane found by our model is applied to the testing data to obtain the classification accuracy. In our model, the label/class

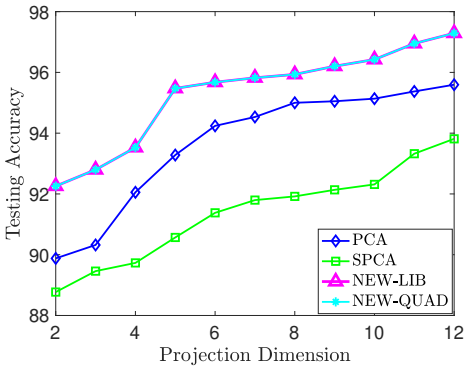
of each unseen data point in the reduced dimension testing data is obtained by the **decision function** in (4.8).

The classification accuracy by the decision function for the best values of hyperparameters C, μ, and γ is presented in Figure 4.7. In this figure, the decision function in (4.8) is only used to find the classification accuracy of the testing data set whose dimension is reduced by our proposed model is solved by LIBSVM [46, 97] and quadprog function. 1NN is applied to find the classification accuracy of the testing data for which the dimension is reduced by PCA and SPCA. Figure 4.7 shows how our model outperforms against other state-of-the-art algorithms and achieves much better results than the methods mentioned in Chapter 2.

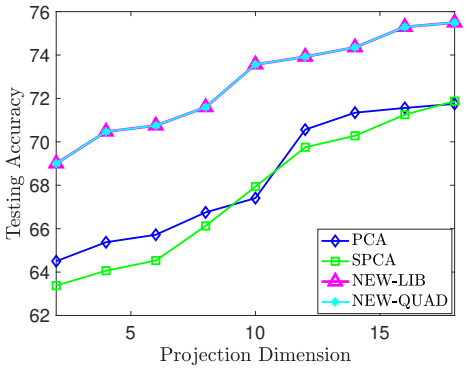


(a) sonar

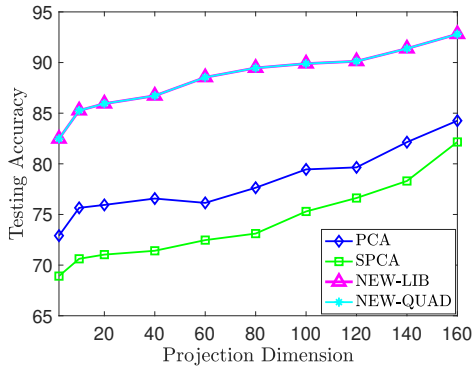
(b) german.numer



(c) wine



(d) svmguide2



(e) dna

Figure 4.7: Classification accuracy on 5 data sets by the decision function in our proposed model as the classification method. 1NN is applied on the testing data whose dimension is reduced by PCA and SPCA.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Summary

In this section, we summarize the mathematical optimization method that proposed in this dissertation to solve dimensionality reduction and classification problems arising in mathematics and data science fields.

In Chapter 2, we reviewed some popularly used dimensionality reduction methods such as Principal Component Analysis (PCA), Linear Discriminate Analysis (LDA), and Supervised Principal Component Analysis (SPCA). Dual and kernel forms of PCA and SPCA were also discussed in this chapter. Two classification algorithms, Support Vector Machine (SVM) and k-Nearest Neighbor (kNN) were also reviewed. Moreover, the procedure of solving a convex quadratic optimization problem by LIBSVM and MATLAB built-in function quadprog was explained.

We found that applying dimensionality reduction methods and classification algorithms to high-dimensional data sets separately brings some challenges. We proposed a model that is able to reduce the dimension of the data and to do classification at the same time by integrating PCA with SVM. We also proposed an alternating way to solve the original optimization problem of our proposed model in (3.1) which is non-convex. We solved the subproblem (3.2) that is a convex quadratic optimization problem for which various optimization packages such as LIBSVM and MATLAB built-in function quadprog have been developed. From all numerical simulations based on synthetic data, the proposed method, either as a diminesionality reduction method or a model that does dimensionality reduction and classification

simultaneously is very efficient and robust. In this regard, our proposed model can be a good option to consider solving dimensionality reduction and classification problems.

To handle and classify high-dimensional data with nonlinear and complex structures, we expanded our model to its nonlinear form by employing a nonlinear mapping, Φ, that embeds vector x into some Hilbert space $\mathcal{H}$. Similar to its linear form, the optimization problem of our nonlinear model in (4.1) is also solved in an alternating way. The plots of testing classification results show the superior and satisfactory performance of our nonlinear model both as a dimensionality reduction method and a model that can reduce the dimension of the data and do classification at the same time.

5.2 Future Work

The research in this dissertation is motivated by the investigation for better mathematical optimization models and tools for solving dimensionality reduction and classification problems. We would like to contribute to a diverse spectrum of mathematical questions and challenges related to data science, dimensionality reduction, image classification, text classification, etc. The mathematical optimization is the heart of a wide variety of data science problems. We would like to enhance the ideas of convex and non-convex optimization techniques to broaden the applicability of our research in recent trends and applications in data science.

We would like to continue our current research and work on various topics in data science. The first question for which we would like to find answer is:

Are there other multi-class classification algorithms to utilize in our model instead of the ones depending on binary classification?

We used the One-vs-One strategy in our model to do the multi-class classification which depends on a binary classification procedure. We would like to propose a more

advanced and computationally efficient algorithm that uses better and more efficient multi-class classification methods that are not dependent on the binary classification strategy.

This research focuses on degree-2 polynomial mapping. Implementation for an efficient degree-3 mapping would be an interesting future study. We also would like to investigate other nonlinear mapping functions to expand data vectors. There might be greater flexibility to design the nonlinear mapping functions, as kernel functions are not used.

Our proposed model in Chapter 4 is linear in dimensionality reduction and nonlinear in classification. Applying nonlinear mappings and kernel functions to the dimensionality reduction part can also be considered.

We also would like to develop efficient open access toolboxes so that researchers and students interested in data science can use our research results and apply them in other areas especially in supervised dimensionality reduction.

Mathematical modeling skills, extensive knowledge of mathematical optimization, and synthesis of powerful novel mathematical concepts are required for answering challenges in dimensionality reduction and classification problems of several domains. We would like to continue working on dimensionality reduction and classification problems in data science and other areas of science. The effort on advancing mathematical research on solving the challenges in the domain of data science will be continued.

REFERENCES

[1] R. E. Bellman, Adaptive control processes: a guided tour. Princeton university press, 2015.

[2] L. O. Jimenez and D. A. Landgrebe, "Supervised classification in high-dimensional space: geometrical, statistical, and asymptotical properties of multivariate data," IEEE Trans. Syst. Man, Cybern. Part C (Applications Rev., vol. 28, no. 1, pp. 39–54, 1998.

[3] https://haifengl.wordpress.com/2016/02/29/there-is-no-big-data-in-machine-learning/

[4] http://jntsai.blogspot.com/2015/04/ammai-nonlinear-dimensionality.html

[5] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," Science (80-.)., vol. 290, no. 5500, pp. 2323–2326, 2000.

[6] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," Neural Comput., vol. 15, no. 6, pp. 1373–1396, 2003.

[7] Z. Zhang and H. Zha, "Principal manifolds and nonlinear dimensionality reduction via tangent space alignment," SIAM J. Sci. Comput., vol. 26, no. 1, pp. 313–338, 2004.

[8] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," Science (80-.)., vol. 290, no. 5500, pp. 2319–2323, 2000.

[9] K. Q. Weinberger and L. K. Saul, "Unsupervised learning of image manifolds by semidefinite programming," Int. J. Comput. Vis., vol. 70, no. 1, pp. 77–90, 2006.

[10] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective," Neurocomputing, vol. 300, pp. 70–79, 2018.

[11] S. Sun and C. Zhang, "Adaptive feature extraction for EEG signal classification," Med. Biol. Eng. Comput., vol. 44, no. 10, pp. 931–935, 2006.

[12] I. Guyon and A. Elisseeff, "An introduction to feature extraction," in Feature extraction, Springer, 2006, pp. 1–25.

[13] L. Maaten, E. Postma, and J. V. Herik. "Dimensionality reduction: a comparative." J Mach Learn Res 10, no. 66-71 (2009): 13.

[14] I. K. Fodor, "A survey of dimension reduction techniques." No. UCRL-ID-148494. Lawrence Livermore National Lab., CA (US), 2002.

[15] L. Jing, C. Zhang, and M. K. Ng, "SNMFCA: Supervised NMF-based image classification and annotation," IEEE Trans. image Process., vol. 21, no. 11, pp. 4508–4521, 2012.

[16] R. Wu, Y. Yu, and W. Wang, "Scale: Supervised and cascaded laplacian eigenmaps for visual object recognition based on nearest neighbors," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 867–874.

[17] X. Zhang, N. Guan, Z. Jia, X. Qiu, and Z. Luo, "Semi-supervised projective non-negative matrix factorization for cancer classification," PLoS One, vol. 10, no. 9, p. e0138814, 2015.

[18] X. Chen, L. Wang, J. D. Smith, and B. Zhang, "Supervised principal component analysis for gene set enrichment of microarray data with continuous or survival outcomes," Bioinformatics, vol. 24, no. 21, pp. 2474–2481, 2008.

[19] S. Yu, K. Yu, V. Tresp, H.-P. Kriegel, and M. Wu, "Supervised probabilistic principal component analysis," in Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, 2006, pp. 464–473.

[20] S.-Y. Lee, H.-A. Song, and S. Amari, "A new discriminant NMF algorithm and its application to the extraction of subtle emotional differences in speech," Cogn. Neurodyn., vol. 6, no. 6, pp. 525–535, 2012.

[21] P. Sprechmann, A. M. Bronstein, and G. Sapiro, "Supervised non-negative matrix factorization for audio source separation," in Excursions in Harmonic Analysis, Volume 4, Springer, 2015, pp. 407–420.

[22] M. Vlachos, C. Domeniconi, D. Gunopulos, G. Kollios, and N. Koudas, "Non-linear dimensionality reduction techniques for classification and visualization," in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, 2002, pp. 645–651.

[23] X. Geng, D.-C. Zhan, and Z.-H. Zhou, "Supervised nonlinear dimensionality reduction for visualization and classification," IEEE Trans. Syst. Man, Cybern. Part B, vol. 35, no. 6, pp. 1098–1107, 2005.

[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Advances in neural information processing systems, 2013, pp. 3111–3119.

[25] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[26] H.-P. Chan et al., "Computer-aided classification of mammographic masses and normal tissue: linear discriminant analysis in texture feature space," Phys. Med. Biol., vol. 40, no. 5, p. 857, 1995.

[27] S. Dudoit, J. Fridlyand, and T. P. Speed, "Comparison of discrimination methods for the classification of tumors using gene expression data," J. Am. Stat. Assoc., vol. 97, no. 457, pp. 77–87, 2002.

[28] L. Chen, N. C. Carpita, W. Reiter, R. H. Wilson, C. Jeffries, and M. C. McCann, "A rapid method to screen for cellwall mutants using discriminant analysis of Fourier transform infrared spectra," Plant J., vol. 16, no. 3, pp. 385–392, 1998.

[29] T. Gaber, A. Tharwat, V. Snasel, and A. E. Hassanien, "Plant identification: Two dimensional-based vs. one dimensional-based feature extraction methods," in 10th international conference on soft computing models in industrial and environmental applications, 2015, pp. 375–385.

[30] X. Zhong and D. Enke, "Forecasting daily stock market return using dimensionality reduction," Expert Syst. Appl., vol. 67, pp. 126–139, 2017.

[31] C. Orsenigo and C. Vercellis, "Linear versus nonlinear dimensionality reduction for banks' credit rating prediction," Knowledge-Based Syst., vol. 47, pp. 14–22, 2013.

[32] K. Fukunaga, Introduction to statistical pattern recognition. Elsevier, 2013.

[33] H. Kim, Howland, P. Park, "Dimension reduction in text classification with support vector machines." J. Mach. Learn. Res. 2005, 6, 37–53.

[34] T. Basu, C. Murthy, "Effective text classification by a supervised feature selection approach." In Proceedings of the 2012 IEEE 12th International Conference on Data Mining Workshops, Brussels, Belgium, 10 December 2012; pp. 918–925.

[35] R. Wu, Y. Yu, W. Wang, "Supervised and cascaded laplacian eigenmaps for visual object recognition based on nearest neighbors." In Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 25–27 June 2013; pp. 867–874.

[36] X. Chen, L. Wang, J.D. Smith, B. Zhang, "Supervised principal component analysis for gene set enrichment of microarray data with continuous or survival outcomes." Bioinformatics 2008, 24, 2474–2481.

[37] J. Ye , Z. Zhao , M. Wu , J.C. Platt , D. Koller , Y. Singer , S. Roweis,"Discriminative k-means for clustering." Proceedings of the NIPS, 2007b, pp. 1649–1656.

[38] D. Wang , F. Nie , H. Huang, "Unsupervised feature selection via unified trace ratio formulation and k-means clustering (track)." Proceedings of the ECML PKDD, 2014, pp. 306–321.

[39] H. Hotelling, "The relations of the newer multivariate statistical methods to factor analysis," Br. J. Stat. Psychol., vol. 10, no. 2, pp. 69–79, 1957.

[40] I. T. Jolliffe, Principal Component Analysis. 2ed., Springer, 2002.

[41] R. A. Fisher, "The use of multiple measurements in taxonomic problems," Annals of eugenics, vol. 7, no. 2, pp. 179–188, 1936.

[42] B. Scholkopft and K.-R. Mullert, "Fisher discriminant analysis with kernels," Neural networks signal Process. IX, vol. 1, no. 1, p. 1, 1999.

[43] E. Barshan, A. Ghodsi, Z. Azimifar, and M. Z. Jahromi, "Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds," Pattern Recognit., vol. 44, no. 7, pp. 1357–1371, 2011.

[44] V. Vapnik, The nature of statistical learning theory. Springer science and business media, 2013.

[45] R. O. Duda, P. E. Hart, and D. G. Stork, Pattern classification. John Wiley and Sons, 2012.

[46] https://www.csie.ntu.edu.tw/∼ cjlin/libsvm/

[47] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition, 1991, pp. 586–587.

[48] R. Huber, H. Ramoser, K. Mayer, H. Penz, and M. Rubik, "Classification of coins using an eigenspace approach," Pattern Recognit. Lett., vol. 26, no. 1, pp. 61–75, 2005.

[49] A. M. Posadas et al., "Spatialtemporal analysis of a seismic series using the principal components method: The Antequera Series, Spain, 1989," J. Geophys. Res. Solid Earth, vol. 98, no. B2, pp. 1923–1932, 1993.

[50] W. Müller, T. Nocke, and H. Schumann, "Enhancing the visualization process with principal component analysis to support the exploration of trends," in Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation-Volume 60, 2006, pp. 121–130.

[51] C.G. Thomas, R.A. Harshman and R.S. Menon," Noise reduction in bold-based fmri using component analysis," Neuroimage 17(3) (2002), 1521–1537. doi10.1006/nimg.2002.1200.

[52] A. L. Price, N. J. Patterson, R. M. Plenge, M. E. Weinblatt, N. A. Shadick, and D. Reich, "Principal components analysis corrects for stratification in genomewide association studies," Nat. Genet., vol. 38, no. 8, pp. 904–909, 2006.

[53] W. Ku, R. H. Storer, and C. Georgakis, "Disturbance detection and isolation by dynamic principal component analysis," Chemom. Intell. Lab. Syst., vol. 30, no. 1, pp. 179–196, 1995.

[54] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum principal component analysis," Nat. Phys., vol. 10, no. 9, pp. 631–633, 2014.

[55] K. Pearson, "LIII. On lines and planes of closest fit to systems of points in space," London, Edinburgh, Dublin Philos. Mag. J. Sci., vol. 2, no. 11, pp. 559–572, 1901.

[56] G. James, D. Witten, T. Hastie, and R. Tibshirani, An Introduction to Statistical Learning with Applications in R. Springer 2017.

[57] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," Neural Comput., vol. 10, no. 5, pp. 1299–1319, 1998.

[58] K. I. Kim, K. Jung, and H. J. Kim, "Face recognition using kernel principal component analysis," IEEE Signal Process. Lett., vol. 9, no. 2, pp. 40–42, 2002.

[59] Q. Wang, "Kernel principal component analysis and its applications in face recognition and active shape models," arXiv Prepr. arXiv1207.3538, 2012.

[60] A. Lima, H. Zen, Y. Nankaku, C. Miyajima, K. Tokuda, and T. Kitamura, "On the use of kernel PCA for feature extraction in speech recognition," IEICE Trans. Inf. Syst., vol. 87, no. 12, pp. 2802–2811, 2004.

[61] H. Hoffmann, "Kernel PCA for novelty detection," Pattern Recognit., vol. 40, no. 3, pp. 863–874, 2007.

[62] Y. Shiokawa, Y. Date, and J. Kikuchi, "Application of kernel principal component analysis and computational machine learning to exploration of metabolites strongly associated with diet," Sci. Rep., vol. 8, no. 1, pp. 1–8, 2018.

[63] D. Widjaja, C. Varon, A. Dorado, J. A. K. Suykens, and S. Van Huffel, "Application of kernel principal component analysis for single-lead-ECG-derived respiration," IEEE Trans. Biomed. Eng., vol. 59, no. 4, pp. 1169–1176, 2012.

[64] http://www.cis.jhu.edu/ cshen/html/PublishSwissRoll.html

[65] N. Aronszajn, Theory of reproducing kernels. Transactions of the American Mathematical Society 68., 1950.

[66] J. Shawe-Taylor and N. Cristianini, Kernel methods for pattern analysis. Cambridge university press, 2004.

[67] K. Q. Weinberger, F. Sha, and L. K. Saul, "Learning a kernel matrix for nonlinear dimensionality reduction," in Proceedings of the twenty-first international conference on Machine learning, 2004, p. 106.

[68] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in Advances in neural information processing systems, 1999, pp. 487–493.

[69] A. Shashua, "Introduction to machine learning: Class notes," arXiv Prepr. arXiv0904.3664, 2009.

[70] M. Aizerman, E. Braverman, and L. Rozonoer, "Theoretical foundations of the potential function method in pattern recognition learning," Automation and Remote Control, 25:821837, 1964.

[71] J.-P. Vert, K. Tsuda, and B. Schölkopf, "A primer on kernel methods," Kernel methods Comput. Biol., vol. 47, pp. 35–70, 2004.

[72] A. Müller, "Kernel Approximations for Efficient SVMs (and other feature extraction methods)," Data Sci. Mach. Learn. Resour., 2012.

[73] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in Advances in neural information processing systems, 2018, pp. 8571–8580.

[74] M. E. Tipping, "Sparse kernel principal component analysis," in Advances in neural information processing systems, 2001, pp. 633–639.

[75] M. C. Wu, L. Zhang, Z. Wang, D. C. Christiani, and X. Lin, "Sparse linear discriminant analysis for simultaneous testing for the significance of a gene set/pathway and gene selection," Bioinformatics, vol. 25, no. 9, pp. 1145–1151, 2009.

[76] D. Huang, Y. Quan, M. He, and B. Zhou, "Comparison of linear discriminant analysis methods for the classification of cancer based on gene expression data," J. Exp. Clin. cancer Res., vol. 28, no. 1, p. 149, 2009.

[77] J. Cui and Y. Xu, "Three dimensional palmprint recognition using linear discriminant analysis method," in 2011 second international conference on innovations in bio-inspired computing and applications, 2011, pp. 107–111.

[78] J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "Face recognition using LDA-based algorithms," IEEE Trans. Neural networks, vol. 14, no. 1, pp. 195–200, 2003.

[79] A. Sharma, S. Imoto, and S. Miyano, "A betweenclass overlapping filterbased method for transcriptome data analysis," J. Bioinform. Comput. Biol., vol. 10, no. 05, p. 1250010, 2012.

[80] A. Sharma, S. Imoto, and S. Miyano, "A filter based feature selection algorithm using null space of covariance matrix for DNA microarray gene expression data," Curr. Bioinform., vol. 7, no. 3, pp. 289–294, 2012.

[81] S. Rezzi, D. E. Axelson, K. Héberger, F. Reniero, C. Mariani, and C. Guillou, "Classification of olive oils using high throughput flow 1H NMR fingerprinting with principal component analysis, linear discriminant analysis and probabilistic neural networks," Anal. Chim. Acta, vol. 552, no. 1–2, pp. 13–24, 2005.

[82] A. Tharwat, T. Gaber, Y. M. Awad, N. Dey, and A. E. Hassanien, "Plants identification using feature fusion technique and bagging classifier," in The 1st International Conference on Advanced Intelligent System and Informatics (AISI2015), November 28-30, 2015, Beni Suef, Egypt, 2016, pp. 461–471.

[83] M. Welling, "Fisher linear discriminant analysis. department of computer science, university of toronto," Technical Report, 2005.

[84] F. Pan, G. Song, X. Gan, and Q. Gu, "Consistent feature selection and its application to face recognition," J. Intell. Inf. Syst., vol. 43, no. 2, pp. 307–321, 2014.

[85] A. Tharwat, T. Gaber, A. Ibrahim, and A. E. Hassanien, "Linear discriminant analysis: A detailed tutorial," AI Commun., vol. 30, no. 2, pp. 169–190, 2017.

[86] S. Balakrishnama and A. Ganapathiraju, "Linear discriminant analysis-a brief tutorial," in Institute for Signal and information Processing, 1998, vol. 18, no. 1998, pp. 1–8.

[87] J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "Face recognition using LDA-based algorithms," IEEE Trans. Neural networks, vol. 14, no. 1, pp. 195–200, 2003.

[88] J. Ye, R. Janardan, and Q. Li, "Two dimensional linear discriminant analysis," in Advances in neural information processing systems, 2005, pp. 1569–1576.

[89] H. Yu and J. Yang, "A direct LDA algorithm for high-dimensional data—with application to face recognition," Pattern Recognit., vol. 34, no. 10, pp. 2067–2070, 2001.

[90] A. Sharma and K. K. Paliwal, "A new perspective to null linear discriminant analysis method and its fast implementation using random matrix multiplication with scatter matrices," Pattern Recognit., vol. 45, no. 6, pp. 2205–2213, 2012.

[91] L. Yang, W. Gong, X. Gu, W. Li, and Y. Liang, "Null space discriminant locality preserving projections for face recognition," Neurocomputing, vol. 71, no. 16–18, pp. 3644–3649, 2008.

[92] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," IEEE Trans. Pattern Anal. Mach. Intell., vol. 19, no. 7, pp. 711–720, 1997.

[93] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, "Measuring statistical dependence with Hilbert-Schmidt norms," in International conference on algorithmic learning theory, 2005, pp. 63–77.

[94] R. A. Fisher, "The precision of discriminant functions," Ann. Eugen., vol. 10, no. 1, pp. 422–429, 1940.

[95] M. J. Greenacre, "Theory and applications of correspondence analysis," 1984.

[96] J. L. Alperin, "Local representation theory: Modular representations as an introduction to the local representation theory of finite groups," vol. 11. Cambridge University Press, 1993.

[97] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," ACM Trans. Intell. Syst. Technol., vol. 2, no. 3, pp. 1–27, 2011.

[98] : http://svmlight.joachims.org/

[99] T. Hastie and R. Tibshirani, "Classification by pairwise coupling," in Advances in neural information processing systems, 1998, pp. 507–513.

[100] J. H. Friedman, "Another approach to polychotomous classification," Tech. Report, Stat. Dep. Stanford Univ., 1996.

[101] C. J. C. Burges and B. Schölkopf, "Improving the accuracy and speed of support vector learning machines," Adv. Neural Inf. Process. Syst., vol. 9, pp. 375–381.

[102] W. S. Noble, "Support vector machine applications in computational biology," Kernel methods Comput. Biol., vol. 71, p. 92, 2004.

[103] J. Ghent and J. McDonald, "Facial expression classification using a one-against-all support vector machine," 2005.

[104] F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche, and B. Arnaldi, "A review of classification algorithms for EEG-based brain–computer interfaces," J. Neural Eng., vol. 4, no. 2, p. R1, 2007.

[105] C.-J. Lin, "Formulations of support vector machines: a note from an optimization point of view," Neural Comput., vol. 13, no. 2, pp. 307–317, 2001.

[106] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," IEEE Trans. Electron. Comput., no. 3, pp. 326–334, 1965.

[107] J. Platt, "Fast training of support vector machines using sequential minimal optimization. Advances in Kernel Methods—Support Vector Learning (pp. 185–208)," AJ, MIT Press. Cambridge, MA, 1999.

[108] R.-E. Fan, P.-H. Chen, and C.-J. Lin, "Working set selection using second order information for training support vector machines," J. Mach. Learn. Res., vol. 6, no. Dec, pp. 1889–1918, 2005.

[109] P.-H. Chen, R.-E. Fan, and C.-J. Lin, "A study on SMO-type decomposition methods for support vector machines," IEEE Trans. Neural Networks, vol. 17, no. 4, pp. 893–908, 2006.

[110] C.-J. Lin, "Asymptotic convergence of an SMO algorithm without any assumptions," IEEE Trans. Neural networks, vol. 13, no. 1, pp. 248–250, 2002.

[111] S. Boyd and L. Vandenberghe, Convex optimization. Cambridge university press, 2007.

[112] B. Ghojogh, F. Karray, and M. Crowley, "Eigenvalue and generalized eigenvalue problems: Tutorial," arXiv Prepr. arXiv1903.11240, 2019.

[113] B. N. Parlett, The symmetric eigenvalue problem. SIAM, 1998.

[114] Y. Saad, Numerical methods for large eigenvalue problems. Manchester University Press, 1992.

[115] T. T. Ngo, M. Bellalij, and Y. Saad, "The trace ratio optimization problem," SIAM Rev., vol. 54, no. 3, pp. 545–569, 2012.

[116] E. Kokiopoulou, J. Chen, and Y. Saad, "Trace optimization and eigenproblems in dimension reduction methods," Numer. Linear Algebr. with Appl., vol. 18, no. 3, pp. 565–602, 2011.

[117] https://www.eric-kim.net/eric-kim-net/posts/1/kerneltrick.html

[118] Y.W. Chang, C.J. Hsieh, K.W. Chang, M.Ringgaard, and C.J. Lin, “Training and testing low-degree polynomial data mappings via linear SVM." Journal of Machine Learning Research 11, no. 4 (2010).

BIOGRAPHICAL STATEMENT

Faezeh Soleimani was born in Kermanshah, Iran, in 1989. She received her Bachelor of Science degree in Applied Mathematics from Razi University of Kermanshah, Iran in 2011. She graduated with her first Master of Science degree in Applied Mathematics-Operations Research from Departments of Mathematics and Computer Science, Kharazmi University of Tehran, Iran in 2014. She joined the Ph.D. program in Mathematics at the University of Texas at Arlington in the fall of 2017. She received her second Master of Science degree in Data Science from Department of Mathematics, The University of Texas at Arlington in 2020.

Faezeh's research interest includes numerical linear algebra, numerical analysis, mathematical optimization, scientific computing, machine learning, data mining, and artificial intelligence. She wants to extend her research skills to solve the challenges in the field of data science.