Aalto University
School of Science
Department of Information and Computer Science
Degree Programme of Computer Science and Engineering

Zhanxing Zhu

Supervised Distance Preserving Projection for Dimensionality Reduction

Master's thesis

Espoo, 10th October 2011

Supervisor: Prof. Olli Simula

Instructor: D.Sc.(Tech.) Francesco Corona

Aalto-yliopisto
Teknillinen korkeakoulu

Aalto University School of Science Department of Information and Computer Science Degree Programme of Computer Science and Engineering	ABSTRACT OF MASTER'S THESIS

Author:	Zhanxing Zhu

Title:	Supervised Distance Preserving Projection for Dimensionality Reduction

Number of pages:	ix + 53	Date:	10th October 2011	Language:	English

Professorship: T-61 Computer and Information Science

Supervisor:	Prof. Olli Simula
Instructor:	D.Sc.(Tech.) Francesco Corona

Abstract:

Facing with high-dimensional data, dimensionality reduction is an essential technique for overcoming the "curse of dimensionality" problem. This work focuses on supervised dimensionality reduction, especially for regression tasks. The goal of dimensionality reduction for regression is to learn a low-dimensional representation of the original high-dimensional data such that this new representation leads to accurate regression predictions.

Motivated by continuity preservation, we propose a novel algorithm for supervised dimensionality reduction named Supervised Distance Preserving Projection (SDPP). In order to preserve the continuity in the low-dimensional subspace, we resort to considering the *local* geometrical structure of the original input space and response space. Inside a neighborhood of each point in the input space, the optimization criterion of SDPP tries to minimize the difference between the distances of the projected covariates and distances of the responses. Consequently, this minimization of distance differences leads to the effect that the local geometrical structure of the low-dimensional subspace optimally *matches* the geometrical characteristics of the response space. The local match not only facilitates an accurate regressor design but also uncovers the necessary information for visualization. Different optimization schemes are proposed for solving SDPP efficiently. Moreover, the parametric mapping we learned can easily handle the out-of-sample data points. A kernelized version of SDPP is derived for nonlinear data. An intuitive extension of SDPP is also presented for classification tasks.

We compare the performance of our method with state-of-the-art algorithms on both synthetic and real-world data. These comparisons show the superiority of our approach on the task of dimensionality reduction for regression and classification.

Keywords:	Supervised dimensionality reduction, regression, classification, optimization, kernel.

Acknowledgments

Since I entered Aalto University School of Science in September 2009, my time has been filled with a wealth of memorable and precious experiences. I have been feeling honored to be surrounded by many truly inspirational and friendly people. Foremost amongst these people have been my supervisor, Prof. Olli Simula and my instructor, Docent Francesco Corona. I would like to thank Prof. Olli Simula for his supervision and for providing me with great working conditions in our department. Francesco, who collaborated much with me in the thesis work, has been a friend and teacher, confidant and ally, and an instructor of the highest calibre. I will continue to stagger towards the standards of excellence that he has laid—thank you. I would also like to thank Dr Zhirong Yang who has always been helpful and a valuable source of perspectives. The close collaboration between us has planted deeper understanding of machine learning in my mind.

I would like to thank all the members in our research group—the Environmental and Industrial Machine Learning (EIML) group. The academic discussions together with many jokes made the group such a wonderful place. Thank you to: Amaury Lendasse (Momo), Federico Montesino Pouzols, Yoan Miche, Elia Liitiäinen, Emil Eirola, Qi Yu, Dusan Sovilj, Mark van Heeswijk, Laura Kainulainen and Ajay Ramaseshan etc. I would also like to thank the two Chinese Ph.D students in my office B307, He Zhang and Xi Chen, who are always helpful and kind-hearted.

I am grateful to our department for accepting me as a member of Honors Program and for the valuable experiences and opportunities I have been provided. I sincerely acknowledge the Finnish Cultural Foundation for providing me the grant to support my master thesis.

I want to thank Chao Wang, Chiwei Wang and Hotloo for their accompanying, during the two years' study in Macadamia program. I will always remember the golden time when we had lunch and insightful discussions together.

There are my friends who have kept me smiling during the stay in Otaniemi. I thank: Rong Wang, Ying Wu, Bo Pang, Jingsong Song, Sheng Gao, Jie Su and so many others who I am sure I have left out. Rong Wang has been, and continues to be a pillar of friendship, my best friend and greatest supporter in my study and daily life—much appreciation goes to him.

I give my thanks to my lovely girlfriend, Ming. I am so grateful for her love and

support during these years, which made me go through the freezing winters in Finland. We will succeed in the end—I believe firmly.

Lastly but the most importantly, I thank my family, including my parents and older brother, for their support, endless love and encouragement over these years. Without them none of this would have been possible.

Zhanxing Zhu
Otaniemi, Espoo, Sep. 23rd, 2011

Contents

Acknowledgements ii

Abbreviations vi

Notation vii

List of Figures viii

List of Tables ix

1 Introduction 1
- 1.1 Dimensionality Reduction . . . 2
  - 1.1.1 Unsupervised Dimensionality Reduction . . . 3
  - 1.1.2 Supervised Dimensionality Reduction . . . 4
- 1.2 Highlights and Organization of the Thesis . . . 5
  - 1.2.1 Highlights of the Thesis . . . 5
  - 1.2.2 Thesis Structure . . . 6

I Theory 8

2 State-of-the-art Approaches for Dimensionality Reduction in Regression 9
- 2.1 Partial Least Squares (PLS) . . . 9
- 2.2 Supervised Principal Component Analysis (SPCA) . . . 11
- 2.3 Kernel Dimensionality Reduction (KDR) . . . 13
- 2.4 Comparison of PLS, SPCA and KDR . . . 14
  - 2.4.1 Toy example one: plane . . . 15
  - 2.4.2 Toy example two: "parity" . . . 15

3 Supervised Distance Preserving Projection (SDPP) 19
- 3.1 Criterion . . . 19
- 3.2 Optimization of SDPP . . . 20
  - 3.2.1 Semidefinite Programming Formulation for SDPP . . . 21
  - 3.2.2 SQLP Formulation for SDPP . . . 23
  - 3.2.3 Conjugate-Gradient Optimization for SDPP . . . 25
- 3.3 Neighborhood Selection Strategies for SDPP . . . 27

3.3.1 Not-enclosing neighborhoods . . . 27
3.3.2 Enclosing neighborhoods . . . 28
3.3.3 Evaluation by A Toy Example . . . 30
3.4 Kernel extension of SDPP . . . 32

II Experiments **34**

4 Experimental Evaluation **35**
4.1 SDPP for Regression . . . 35
4.1.1 Curve line . . . 35
4.1.2 Predicting rotation angles on an image manifold . . . 37
4.1.3 Regression on UCI data sets . . . 39
4.2 SDPP for Classification . . . 41
4.2.1 Tai Chi . . . 41
4.2.2 Classification on UCI data sets . . . 42

5 Conclusions **47**

Abbreviations

DR	Dimensionality reduction
SDPP	Supervised Distance Preserving Projection
KSDPP	Kernel Supervised Distance Preserving Projection
PCA	Principal component analysis
LLE	Local linear embedding
ISOMAP	Isometric Mapping
MVU	Maximum variance unfolding
FDA	Fisher discriminant analysis
GDA	Generalized discriminant analysis
PLS	Partial least squares
KPLS	Kernel partial least squares
CCA	Canonical correlation analysis
SDR	Sufficient dimensionality reduction
KDR	Kernel dimension reduction
RKHS	Reduced Kernel Hilbert Space
HSIC	Hilbert-Schmidt independence criterion
SPCA	Supervised principal component analysis
PCA	Principle component analysis
NIPALS	Nonlinear iterative partial least squares
SIR	Sliced inverse regression
pHd	Principal Hessian directions
SAVE	Sliced average variance estimation
PSD	Positive semidefinite
QSDP	Convex quadratic semidefinite programming
SDP	Semidefinite Programming
SQLP	Semidefinite quadratic linear programming
CG	Conjugate gradient
SVD	Singular value decomposition

Notation

$\mathcal{X}$	Input space
$\mathcal{Y}$	Response space
$\mathcal{Z}$	Low-dimensional subspace
n, n_t	Number of training points and test points
d	The dimensionality of $\mathcal{X}$
l	The dimensionality of $\mathcal{Y}$
r	The dimensionality of $\mathcal{Z}$
$\mathbf{I}$	Identity matrix
$\mathbf{1}$	A vector with all ones
$\mathbf{0}$	A vector with all zeros
$\mathbf{e}_i$	i-th basis vector
$\mathbf{X}$	$n \times d$ matrix of input covariates, each row is a data sample
$\mathbf{Y}$	$n \times l$ matrix of output responses
$\mathbf{Z}$	$n \times r$ matrix of low-dimensional projections
$\mathbf{X}_{ij}$	Entry of a matrix: i-th row, j-th column
$\mathbf{x}_i$	i-th input
$\mathbf{y}_i$	i-th output
$\mathbf{z}_i$	i-th low-dimensional representation
X_i	i-th feature of $\mathbf{x}$, with $i = 1, \ldots, d$
$\mathbf{W}$	Projection matrix in $\mathbb{R}^{d \times r}$
$\mathbf{D}$	Squared distance matrix in the low-dimensional subspace
$\mathbf{\Delta}$	Squared distance matrix in the response space
$N(\mathbf{x})$	The neighborhood of $\mathbf{x}$
k	Number of the nearest neighbors
$\mathcal{J}(\mathbf{W})$	The objective function with respect to $\mathbf{W}$
$\nabla_{\mathbf{W}}$	The gradient with respect to $\mathbf{W}$
$k(\cdot)$	Kernel function
$\mathbf{K}$	Kernel matrix
$\mathbf{K}^c$	Centered kernel matrix
$\mathbf{K}_{\mathbf{x}}$	$n \times n$ kernel matrix for training set in $\mathcal{X}$
$\mathbf{K}_{\mathbf{y}}$	$n \times n$ kernel matrix for training set in $\mathcal{Y}$
$\mathbf{K}_{\mathbf{z}}$	$n \times n$ kernel matrix for the low-dimensional subspace
$\mathbf{K}_{\text{test}}$	$n_t \times n$ kernel matrix between the test points and training points
$E(\cdot)$	Expectation operator
$V(\cdot)$	(Co)variance operator

List of Figures

1.1 Curse of dimensionality: 10 points to fill the one, two and three-dimensional space, $[0,1]^d$, where $d = 1, 2, 3$. 2
1.2 A two-dimensional projection of a person's faces, from [51]. 4
1.3 Thesis structure. 7

2.1 Projection comparison on linear data. 16
2.2 Projection comparison on "parity" data. 17

3.1 Illustration of SDPP 21
3.2 Eigengap plots for linear, "parity"and curve data sets. 25
3.3 Not-enclosing neighborhoods. 28
3.4 Enclosing neighborhoods. 29
3.5 Projection results based on not-enclosing neighborhoods. 31
3.6 Projection results based on enclosing neighborhoods. 32

4.1 Projection comparison on curve line data set. 36
4.2 Four sample images of rotating ducks 37
4.3 One-dimensional embedding of the ducks data set. 38
4.4 Two-dimensional embedding of the ducks data set. 39
4.5 Tai Chi with two classes Yin and Yang. 41
4.6 Simulation of Tai Chi model. 42
4.7 Two-dimensional projection of the Tai Chi data for the test set. 42
4.8 Two-dimensional projection of the Glass data for the test set. 44
4.9 Two-dimensional projection of the Segmentation data for the test set. . 45
4.10 Two-dimensional projection of SDPP on the Isolet data for the test set. 45
4.11 Two-dimensional projection of FDA on the Isolet data for the test set. 46
4.12 Two-dimensional projection of SPCA on the Isolet data for the test set. 46

List of Tables

4.1 Description of the data sets for regression tasks. 35
4.2 RMSE ("mean $\pm$ std") on the test sets of three data, Servo, Tecator-fat and Auto-price. 40
4.3 Description of data sets for classification tasks. 43
4.4 Classification accuracies by percentage on the test sets of data, Glass, Segmentation and Isolet. 43

Chapter 1

Introduction

Scientists and engineers often work with a large amount of high-dimensional data, such as images, texts, global climate patterns or human genes. One of the problems when analyzing these data is that the number of dimensions of these data can be extremely large—several thousands, and in some cases, even millions. Such a high number of dimensions may be enough to either make storing the data in memory infeasible, or analyzing and exploring the data difficult. The latter problem is more common, because in the area of machine learning there are many models and algorithms of practical interest that do not perform efficiently if we start with a very large number of dimensions compared with only a few data samples.

This induces the so-called "curse of dimensionality" [6] that has pestered researchers in a wide range of fields for many years. For any data, as the number of dimensions increases, the number of data samples required to learn a specific model increases at an exponential rate, which makes all but the most trivial manipulations and analysis impractical. A dilemma occurs because naturally we would like to be able to use as many informative dimensions as are available to us—however, the "curse" is that since more dimensions are collected, we spend dramatically more time trying to make sense of them.

Here we show a simple example of the "curse" under the task that we try to get representative samples of the set $[0, 1]^d$, where d is the dimension of the data, see Figure 1.1. In the one-dimensional case, this is just an interval—and 10 random samples are enough to obtain a good representation of this interval (for example, it is very unlikely that all samples will be on one half of the interval). When we move to the case with two dimensions, 10 samples are reasonable, but we can see that it is now more likely that we will leave some areas unexplored. Once we get to three dimensions, we notice that these 10 samples are not enough any more and begin to get rather sparse in this cube. This is also called "empty space phenomenon" [46]. It is easy to imagine the problem with several thousands of dimensions—by linearly increasing the number of dimensions, the number of points required should be increased by an exponential rate to fill the new space.

In addition, some high-dimensional data usually contain noise. The inclusion of

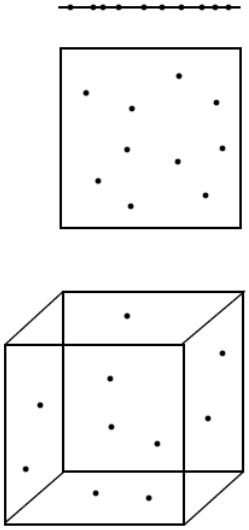


Figure 1.1: Curse of dimensionality: 10 points to fill the one, two and three-dimensional space, $[0,1]^d$, where $d = 1, 2, 3$.

unnecessary variables, called noisy variables, could complicate or even ruin the data structure [15]. Fan and Fan [17] have theoretically proved that if the signal-to-noise ratio is too small, linear discriminant rules in classification tasks become no better than random guessing due to the noise accumulation in the estimation of the population mean vectors.

Moreover, the high-dimensional nature of data is often simply a product of its representation. In many instances data dimensions are redundant and entirely correlated with some combination of other dimensions within the same data set. In these cases, although the retrieved data seem to exhibit a naturally high dimension, it is actually constrained to a lower dimensional representation of the measurement space.

1.1 Dimensionality Reduction

Based on these considerations on high-dimensional data, it is necessary to find an approximation (or representation) in low-dimensional space, which is referred as *dimensionality reduction* (DR). The intent here is that by reducing the number of dimensions, we reduce significantly the time that is needed for our analysis, but, at the same time, we keep as much information as we can about the original data, so that it is a good representation. From this point of view, DR can be a preprocessing step for many data analysis tasks (for example, clustering, classification and regression). That is, instead of performing our analysis with the original data, we work on the low-dimensional representation. Additionally, by reducing the original high-dimensional data into a two or three-dimensional space, DR also serves as a visualization tool for unveiling the intrinsic structure of the original data.

Generally, based on different settings of original data, there are two kinds of DR categories, unsupervised DR and supervised DR. Next, we will present a brief overview of the two kinds of DR.

1.1.1 Unsupervised Dimensionality Reduction

When handling high-dimensional data in unsupervised settings (i.e., the unlabelled data samples), dimensionality reduction is mainly used as a preprocessing step for clustering or visualization of the data. For example, in Figure 1.2, the input consists of many images of a person's face observed under different pose and lighting conditions. These images can be thought of as points in a high-dimensional vector space, with each input dimension corresponding to the brightness of one pixel in the image. There are not any labels for these images and we do not know any information of the pose and lighting conditions. Although the input dimensionality may be quite high (e.g., 4096 for these 64 by 64 pixels images), the perceptually meaningful structure of these images has much fewer independent degrees of freedom. Within the 4096-dimensional input space, all of the images lie on an intrinsically three-dimensional space, which can be represented by two pose variables plus an azimuthal lighting angle. The goal of unsupervised dimensionality reduction is to discover, given only the unlabelled high-dimensional inputs, low-dimensional representations, with coordinates that capture the informative structure of the data, and to visualize them. Figure 1.2 presents a two-dimensional projection of a person's faces, with a sample of the original input images (red circles) superimposed on all the data points (blue) and horizontal sliders (under the images) representing the third dimension. Each coordinate axis of the embedding correlates highly with one degree of freedom underlying the original data: left-right pose (x-axis), up-down pose (y-axis), and lighting direction (slider position).

Depending on different criteria, much research devotion has focused on the unsupervised learning settings of dimensionality reduction. Generally, there are two criteria used commonly, *distance preservation* and *topology preservation*.

In the first category, all the methods aim to preserve specific distances between the original data and their embeddings. Principal component analysis (PCA), multidimensional scaling (MDS) and its variants, Sammon's nonlinear mapping [44], and curvilinear component analysis [14] try to retain the spatial distances after embedding. The adjective "spatial" indicates that the distance metrics compute the distance only using the coordinates of the two points matter, without regards to any other information like the presence of a submanifold. For instance, Euclidean distance is the most commonly used spatial distance. To reduce the dimensionality of highly folded manifolds, ISOMAP [51] preserves the geodesic distances which are approximated by graph distance based on neighborhood construction. Stemming from mathematical considerations about kernel functions, kernel PCA [45] considers the pairwise distances in feature space. Different from kernel PCA, maximum variance unfolding (MVU) [59] learns a kernel matrix by defining a neighborhood graph on the data and preserving pairwise distances in the resulting graph. Concretely, MVU attempts to maximize

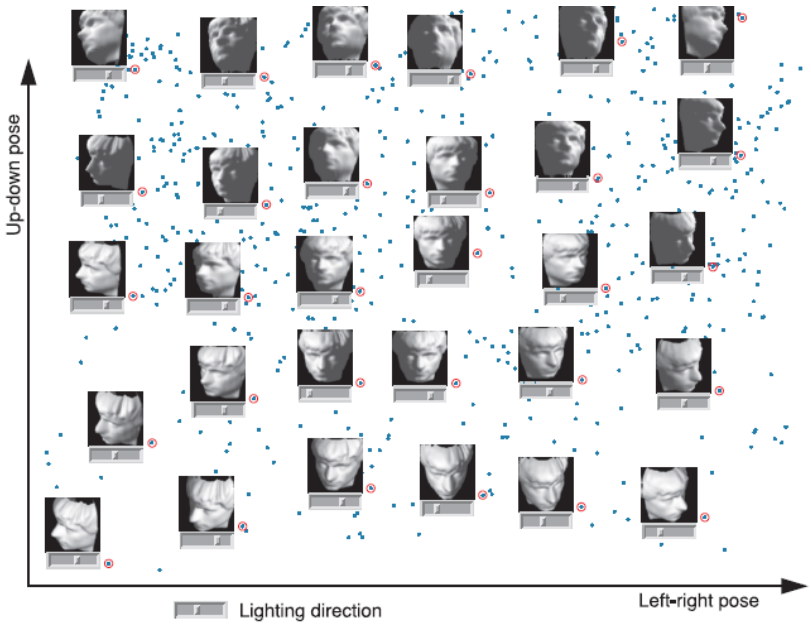


Figure 1.2: A two-dimensional projection of a person's faces, from [51].

the sum of squared Euclidean distances between all data points, under the constraint that the distances inside the neighborhood graph are preserved.

In the second category, the methods reduce the dimensionality by preserving the topology of data rather than their pairwise distances. Self-organizing map (SOM) [27] implements a spatially ordered mapping of a high-dimensional data onto a predefined lattice based on competitive learning of artificial neurons. In contrast with SOM using a predefined lattice, local linear embedding (LLE) [43] and Laplacian eigenmaps [4] make no assumption on the shape and topology of the embedding. Instead, they consider the information contained in data in order to establish the topology of the data set and compute the shape of the embedding accordingly. LLE assumes the local linear structure of the manifold and describes the local properties of the manifold around a data point by considering the data point as a linear combination of the its nearest neighbors. In the low-dimensional representation of the data, LLE tries to retain the reconstruction weights in the linear combination as much as possible. Laplacian eigenmap finds a low-dimensional data representation by preserving local properties of the manifold. The local properties in Laplacian eigenmaps are based on the pairwise distances between nearest neighbors. For a more detailed comparative survey of unsupervised DR techniques, see [34, 29].

1.1.2 Supervised Dimensionality Reduction

In supervised settings, each data sample is labelled which can guide the search of low-dimensional space. In classification tasks, the labels are valued by discrete numbers to show that which data samples belong to the same class. This supervised dimensionality reduction framework with class-labelled information is often referred as discriminative learning. The most widely investigated method is Fisher discrim-

inant analysis (FDA) [8], together with its kernelized form kernel FDA [36] and its generalized version GDA [3]. All of them try to find projection axes for a good separation of classes based on the ratio of between-class and within-class covariance. Moreover, some metric learning algorithms [21, 58, 62] search an appropriate metric to maximize the discriminative power in the new metric space.

In some cases, the labels of data samples take continuously varying real values (referred as output responses) in the task of regression. Thus, in this setting of supervised dimensionality reduction for regression, the aim is to find a low-dimensional representation of the input covariates that can lead regression prediction as accurately as possible.

Partial least square (PLS) [60, 41] is a classic dimensionality reduction method for regression, which constructs orthogonal latent components by maximizing the covariance between the input covariates and output responses. In spirit, PLS is similar to Canonical correlation analysis (CCA) where latent components with maximal correlation are extracted. In order to handle the nonlinear case, its kernelized version kernel PLS was developed in [42]. Another way to achieve dimensionality reduction for regression tasks is known as sufficient dimensionality reduction (SDR) [31, 63, 20]. To preserve the information for the purpose of prediction, SDR finds the subspace (or central subspace) bases such that the projection results the conditional independence of output responses and the original covariates. Kernel dimension reduction (KDR) [20] demonstrates itself as an effective approach for SDR that maximizes the conditional dependency by a positive definite ordering of the expected covariance operators in the so-called probability-determining Reduced Kernel Hilbert Space (RKHS). Recently, based on Hilbert-Schmidt independence criterion (HSIC) [23], Barshan et.al [2] proposed another supervised dimensionality reduction method called supervised principal component analysis (SPCA). SPCA attempts to create the principal components with maximum dependency on the output responses. The optimization of SPCA is solved by eigen-decomposition of weighted covariance matrix of low-dimensional representations enhanced by the kernel of output responses.

1.2 Highlights and Organization of the Thesis

1.2.1 Highlights of the Thesis

In this thesis, we consider dimensionality reduction under the supervised setting, especially for regression. Our goal is to learn a low-dimensional representation of the original high-dimensional data such that this new representation can lead to the regression predictions (or classification) as accurately as possible. Also the low-dimensional representation can help us to reveal the relationship between input covariates and output responses. Different from unsupervised dimensionality reduction, in supervised setting, the output responses (or labels) provide us a guide to search for the low-dimensional representation. Therefore, we incorporate the supervised information into our objective function to find a proper parametric mapping for dimensionality reduction. Inspired by the usage of neighborhood graph in many manifold learning

algorithms, we consider the distance preservation locally.

Summing up these considerations, we propose a novel algorithm for supervised dimensionality reduction, named as Supervised Distance Preserving Projection (SDPP). Briefly, SDPP works as follows. Inside a neighborhood, SDPP tries to minimize the difference between the distances of projected covariates and distances of the responses. Consequently, this minimization of distance difference leads to the effect that the local geometric structure of the input subspace "mimics" the geometric characteristics of the response space. This not only facilitates the efficient regressor design but also uncovers the necessary information for visualization and accurate response prediction. This is our main contribution for solving the problem of dimensionality reduction under the supervised setting. Moreover, the parametric mapping we learned can easily handle the out-of-sample data points. Also kernelized version of SDPP is derived for nonlinear data. An intuitive extension of SDPP is also presented suitable for classification tasks.

In this thesis, we formulate this criterion and present the interpretation behind this criterion. Different optimization schemes are applied to solve the problem of minimizing this criterion. We compare the performance of our method with state-of-the-art algorithms on both synthetic and real-world data. These comparisons show many superiorities of our approach on the task of supervised dimensionality reduction.

1.2.2 Thesis Structure

Generally, the thesis consists of two parts: the first part involves the theory contents of the thesis to overview the state-of-the-art approaches for dimensionality reduction for regression and present the novel method SDPP we have proposed; the second part contains the experimental evaluation of our method compared with other algorithms. Figure 1.3 provides an overview of the thesis structure.

In Part I (Theory), Chapter 2 describes related work on supervised dimensionality reduction. We analyze three representative approaches for supervised dimensionality reduction, PLS, SPCA and KDR and compare them with each other. Chapter 3 presents the criterion of our method SDPP and three optimization schemes for SDPP. Additionally, different neighborhood selection strategies for SDPP are shown in this chapter. To handle the nonlinear data, the kernel version of SDPP is also derived. Toy examples are provided through Chapter 2 and Chapter 3 to show the performance of these approaches.

In Part II (Experiments), Chapter 4 conducts various experiments on both synthetic and real-world data to demonstrate the effectiveness on supervised dimensionality reduction compared with other state-of-the-art methods.

Lastly, in Chapter 5, we conclude the thesis and propose some further research directions based on our current study.

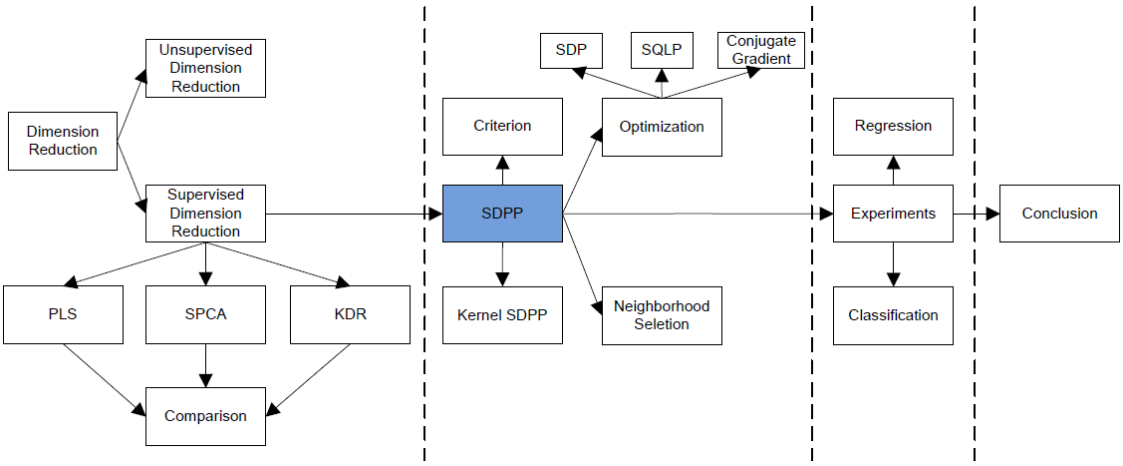


Figure 1.3: Thesis structure.

Part I

Theory

Chapter 2

State-of-the-art Approaches for Dimensionality Reduction in Regression

In this chapter, the main scope is to describe three representative methods for supervise dimensionality reduction in regression, partial least squares, supervised PCA and kernel dimensionality reduction. Afterwards, we analyze the three methods in detail and compare them with each other.

In the following, unless stated otherwise, we always suppose to have n data points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \in \mathbb{R}^d$ and their multiple responses $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\} \in \mathbb{R}^l$ stored in zero-mean matrices $\mathbf{X}_{n\times d}$ and $\mathbf{Y}_{n\times l}$.

2.1 Partial Least Squares (PLS)

PLS is a technique for modelling linear relationship between a set of input covariates and output responses. PLS decomposes these matrices into

$$\begin{aligned} \mathbf{X} &= \mathbf{Z}\mathbf{A}^T + \mathbf{E} \\ \mathbf{Y} &= \mathbf{T}\mathbf{B}^T + \mathbf{F}, \end{aligned} \tag{2.1}$$

where $\mathbf{Z}$ and $\mathbf{T}$ are $n\times r$ are the score matrices, $\mathbf{A}$ and $\mathbf{B}$ are the loading matrices and $\mathbf{E}$ and $\mathbf{F}$ are the residuals. PLS aims to find the projection matrices $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_r]$ and $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_r]$ such that the covariances between the projected inputs and responses are maximized.

$$\max_{\mathbf{w}_i, \mathbf{u}_i} [\mathrm{Cov}(\mathbf{z}_i, \mathbf{t}_i)]^2 = \max_{\mathbf{w}_i, \mathbf{u}_i} [\mathrm{Cov}(\mathbf{X}\mathbf{w}_i, \mathbf{Y}\mathbf{u}_i)]^2, \tag{2.2}$$

where $\mathrm{Cov}(\mathbf{z}_i, \mathbf{t}_i) = \mathbf{z}_i^T \mathbf{t}_i / n$ denotes the sample covariance between score vectors $\mathbf{z}$ and $\mathbf{t}$. PLS is implemented as an iterative procedure. In each iteration, after extraction of $\mathbf{z}$ and $\mathbf{t}$, matrices $\mathbf{X}$ and $\mathbf{Y}$ are deflated by subtracting the information contained in the derived vectors $\mathbf{z}$ and $\mathbf{t}$ [41]. Depending on the form of deflation, several variations of PLS have been proposed. In most of the proposed variations, $\mathbf{X}$ and $\mathbf{Y}$ are deflated

separately. This usually leads to an iterative solution.

In its original form, NIPALS [60] algorithm is used, which is a robust procedure for solving singular value decomposition problems. NIPALS starts with a random initialization of the score vector $\mathbf{t}$ and repeats following steps until convergence.

1. $\mathbf{w} = \mathbf{X}^T\mathbf{t}/(\mathbf{t}^T\mathbf{t})$, $\mathbf{w} = \mathbf{w}/\|\mathbf{w}\|$
2. $\mathbf{z} = \mathbf{X}\mathbf{w}$
3. $\mathbf{u} = \mathbf{Y}^T\mathbf{z}/(\mathbf{z}^T\mathbf{z})$, $\mathbf{u} = \mathbf{u}/\|\mathbf{u}\|$
4. $\mathbf{t} = \mathbf{Y}\mathbf{u}$

Note that $\mathbf{t} = \mathbf{y}$ if $l = 1$, i.e., output response $\mathbf{Y}$ is a one-dimensional vector $\mathbf{y}$. After the extraction of these vectors, $\mathbf{w}$, $\mathbf{z}$, $\mathbf{u}$ and $\mathbf{t}$, we deflate $\mathbf{X}$, $\mathbf{Y}$ matrices: $\mathbf{X} \leftarrow \mathbf{X} - \mathbf{z}\mathbf{z}^T\mathbf{X}$, $\mathbf{Y} \leftarrow \mathbf{Y} - \mathbf{t}\mathbf{t}^T\mathbf{Y}$. The PLS is an iterative process as mentioned before; i.e. after extraction of one component the NIPALS algorithm starts again using the deflated matrices $\mathbf{X}$ and $\mathbf{Y}$. Thus we can achieve the sequence of the components upto the point when the rank of $\mathbf{X}$ is reached.

Furthermore, there is another variant of PLS called PLS-SB [57], where deflations are performed on the cross-product matrix $\mathbf{X}\mathbf{Y}^T$ instead of separate deflations of $\mathbf{X}$ and $\mathbf{Y}$. This scheme leads to extraction of all latent vectors at once by solving an eigenvalue problem directly:

$$\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X}\mathbf{w} = \lambda\mathbf{w} \tag{2.3}$$

and computing $\mathbf{Z}$ and $\mathbf{T}$ as

$$\begin{aligned} \mathbf{Z} &= \mathbf{X}\mathbf{W} \\ \mathbf{U} &= \mathbf{Y}^T\mathbf{T} \\ \mathbf{T} &= \mathbf{Y}\mathbf{U} \end{aligned} \tag{2.4}$$

It can be shown that the two above-mentioned variants of PLS are equivalent [26].

Kernel PLS

The linear PLS is limited when handling the data sets exhibiting nonlinear behaviours. Kernel PLS (KPLS) [42] is designed for dealing with nonlinear data based on a mapping of the original data by means of a nonlinear function to a new representation where PLS is applied.

KPLS maps the original $\mathcal{X}$-space (original input space) into a high-dimensional feature space $\mathcal{F}$ corresponding to a reproducing kernel Hilbert space (RKHS), $\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x})$. Denote the Gram matrix $\mathbf{K}$ of the cross dot products between all mapped input data points, $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \langle\boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j)\rangle$, i.e., $\mathbf{K} = \boldsymbol{\Phi}^T\boldsymbol{\Phi}$, where $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1), \ldots, \boldsymbol{\phi}(\mathbf{x}_n)]$. By using kernel trick, kernelized NIPALS algorithm implements as follows.

1. randomly initialize $\mathbf{t}$
2. $\mathbf{z} = \mathbf{K}\mathbf{u}$, $\mathbf{z} = \mathbf{z}/|\mathbf{z}\|$
3. $\mathbf{t} = \mathbf{Y}\mathbf{Y}^T\mathbf{z}$, $\mathbf{t} = \mathbf{t}/|\mathbf{t}\|$
4. repeat step 2-3 until convergence.
5. deflate $\mathbf{K}$ and $\mathbf{Y}$. $\mathbf{K} \leftarrow \left(\mathbf{\Phi} - \mathbf{z}\mathbf{z}^T\mathbf{\Phi}\right)^T \left(\mathbf{\Phi} - \mathbf{z}\mathbf{z}^T\mathbf{\Phi}\right)$, and $\mathbf{Y} \leftarrow \mathbf{Y} - \mathbf{t}\mathbf{t}^T Y$.

Similar to the original PLS, the kernelized NIPALS is applied sequentially until the rank of $\mathbf{K}$ is reached.

2.2 Supervised Principal Component Analysis (SPCA)

SPCA [2] is a supervised generalization of classic PCA that has shown effectiveness for regression and classification problems with high-dimensional data. SPCA aims to find the subspace such that the dependence between the projected data and the responses is maximized. The dependence measure is based on the Hilbert-Schmidt Independence Criterion (HSIC) [23, 24]. HSIC is equal to zero if and only if two random variables are independent. An empirical estimation of HSIC is

$$\text{HSIC}(\mathcal{Z}, \mathcal{Y}) = \frac{1}{(n-1)^2}\text{Tr}(\mathbf{K_z}\mathbf{H}\mathbf{K_y}\mathbf{H}), \tag{2.5}$$

where $\mathbf{K_z}$ and $\mathbf{K_y}$ are the kernel matrices for projected covariates and output responses, respectively, and $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$ act as centering matrix, $\mathbf{I}$ is the identity matrix, $\mathbf{1}$ is the vector of all ones with length n and $\text{Tr}(\cdot)$ denotes the trace of a matrix.

In dependence measure HSIC, $\text{Tr}(\mathbf{H}\mathbf{K_z}\mathbf{H}\mathbf{K_y})$, we compute the kernel matrix for the projected inputs as $\mathbf{K_z} = \mathbf{X}\mathbf{W}\mathbf{W}^T\mathbf{X}^T$ by a linear kernel, and the kernel matrix for the responses $\mathbf{K_y} = \mathbf{Y}\mathbf{Y}^T$. Thus, the objective function for SPCA is formulated as

$$\begin{aligned} \text{Tr}(\mathbf{H}\mathbf{K_z}\mathbf{H}\mathbf{K_y}) &= \text{Tr}(\mathbf{H}\mathbf{X}\mathbf{W}\mathbf{W}^T\mathbf{X}^T\mathbf{H}\mathbf{K_y}) && (2.6)\\ &= \text{Tr}(\mathbf{W}^T\mathbf{X}^T\mathbf{H}\mathbf{K_y}\mathbf{H}\mathbf{X}\mathbf{W}). && (2.7) \end{aligned}$$

Since we would like to capture the dependence between the projected inputs and the output responses as much as possible, the maximization of HSIC should be conducted. We also prefer to search for an orthogonal transformation matrix $\mathbf{W}$ so that the the projected features are uncorrelated. Consequently, the constrained optimization problems is

$$\begin{aligned} &\arg\max_{\mathbf{W}} \text{Tr}(\mathbf{W}^T\mathbf{X}^T\mathbf{H}\mathbf{K_y}\mathbf{H}\mathbf{X}\mathbf{W}) \\ &\text{s.t. } \mathbf{W}^T\mathbf{W} = \mathbf{I}. \end{aligned} \tag{2.8}$$

By Rayleigh quotient theorem, the optimization can be easily solved in eigen-decomposition form,

$$(\mathbf{X}^T\mathbf{H}\mathbf{K_y}\mathbf{H}\mathbf{X})\mathbf{w} = \lambda\mathbf{w}. \tag{2.9}$$

Algorithm 1 Supervised PCA

Input: training data matrix $\mathbf{X}$, test data point $\mathbf{x}$, kernel matrix for response $\mathbf{K_y}$ and dimensionality of subspace, r.

Output: projected training data matrix $\mathbf{Z}$ and projected test point $\mathbf{z}$.

1. $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{e}\mathbf{e}^T$;

2. compute the orthogonal basis, $\mathbf{W} \leftarrow$ eigenvectors of $\mathbf{X}^T\mathbf{H}\mathbf{K_y}\mathbf{H}\mathbf{X}$ corresponding to the top r eigenvalues;

3. project training data, $\mathbf{Z} \leftarrow \mathbf{X}\mathbf{W}$;

4. project test data, $\mathbf{z} \leftarrow \mathbf{W}^T\mathbf{x}$.

Concretely, we summarize SPCA procedure in Algorithm 1.

We can observe that SPCA is a supervised generalization of PCA. In the scenario of unsupervised setting, the response information is unknown, and then the kernel matrix $\mathbf{K_y}$ for the response should be identity matrix. Maximization of the dependence between kernel matrix $\mathbf{K_z}$ and the identity matrix is equivalent to preserving the maximal variance among all the observations, which can be seen as classical PCA. The form of $\mathbf{X}^T\mathbf{H}\mathbf{K_y}\mathbf{H}\mathbf{X}$ in the unsupervised setting demonstrates the equivalence,

$$\begin{aligned}\mathbf{X}^T\mathbf{H}\mathbf{K_y}\mathbf{H}\mathbf{X} &= (\mathbf{X}\mathbf{H})^T(\mathbf{X}\mathbf{H}) \\ &= \left(\mathbf{X}(\mathbf{I} - \frac{1}{n}\mathbf{e}\mathbf{e}^T)\right)^T \left(\mathbf{X}(\mathbf{I} - \frac{1}{n}\mathbf{e}\mathbf{e}^T)\right) \\ &= (\mathbf{X} - \boldsymbol{\mu}_{\mathcal{X}})^T(\mathbf{X} - \boldsymbol{\mu}_{\mathcal{X}}) \\ &= \mathrm{Cov}(\mathbf{X}),\end{aligned} \tag{2.10}$$

where $\boldsymbol{\mu}_{\mathcal{X}}$ denotes the mean of $\mathbf{X}$.

Kernel SPCA

The kernelized version of SPCA is also derived in [2] to handle the nonlinear data. The key idea is to assume the projection matrix as a linear combination of the data points mapped into feature space, $\mathbf{W} = \boldsymbol{\Phi}\boldsymbol{\Omega}$, by the representation theory [1]. In this case, the objective function can be written as

$$\begin{aligned}\mathrm{Tr}(\mathbf{W}^T\boldsymbol{\Phi}\mathbf{H}\mathbf{K_y}\mathbf{H}\boldsymbol{\Phi}^T\mathbf{W}) &= \mathrm{Tr}(\boldsymbol{\Omega}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\mathbf{H}\mathbf{K_y}\mathbf{H}\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\Omega}) \\ &= \mathrm{Tr}(\boldsymbol{\Omega}^T\mathbf{K}\mathbf{H}\mathbf{K_y}\mathbf{H}\mathbf{K}\boldsymbol{\Omega}),\end{aligned}$$

under the constraint that

$$\mathbf{W}^T\mathbf{W} = \boldsymbol{\Omega}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\Omega} = \boldsymbol{\Omega}^T\mathbf{K}\boldsymbol{\Omega} = \mathbf{I}.$$

Thus, the optimization problem is formulated as follows:

$$\begin{aligned}&\arg\max_{\boldsymbol{\Omega}} \mathrm{Tr}(\boldsymbol{\Omega}^T\mathbf{K}\mathbf{H}\mathbf{K_y}\mathbf{H}\mathbf{K}\boldsymbol{\Omega}) \\ &\text{s.t. } \boldsymbol{\Omega}^T\mathbf{K}\boldsymbol{\Omega} = \mathbf{I}.\end{aligned} \tag{2.11}$$

This formulation can be solved by a generalized eigen-decomposition procedure,

$$(\mathbf{KHK_yHK})\boldsymbol{\omega} = \lambda \mathbf{K}\boldsymbol{\omega}, \tag{2.12}$$

where $\boldsymbol{\omega}$ is one column of $\boldsymbol{\Omega}$. The procedure of kernel SPCA (KSPCA) is summarized in Algorithm 2.

Algorithm 2 Kernel supervised PCA

Input: kernel matrix of training data $\mathbf{K}$, kernel matrix of test data $\mathbf{K}_{\text{test}}$, kernel matrix for response $\mathbf{L}$ and dimensionality of subspace, r.

Output: projected training data matrix $\mathbf{Z}$ and projection of test point $\mathbf{z}$.

1. $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{e}\mathbf{e}^T$;

2. compute the orthogonal basis, $\boldsymbol{\Omega} \leftarrow$ generalized eigenvectors of $(\mathbf{KHK_yHK}, \mathbf{K})$ corresponding to the top r eigenvalues;

3. project training data, $\mathbf{Z} \leftarrow [\boldsymbol{\Phi}^T\boldsymbol{\Phi}]\boldsymbol{\Omega} = \mathbf{K}\boldsymbol{\Omega}$;

4. project test data, $\mathbf{z} \leftarrow [\boldsymbol{\Phi}^T\boldsymbol{\phi}(\mathbf{x})]\boldsymbol{\Omega} = \boldsymbol{\Omega}^T\mathbf{K}_{\text{test}}$.

2.3 Kernel Dimension Reduction (KDR)

KDR [19, 20] is a representative approach of sufficient dimensionality reduction (SDR). SDP tries to find an orthogonal transformation $\mathbf{W}$ such that $\mathcal{Y}$ and $\mathcal{X}$ are conditionally independent provided the subspace $\mathbf{W}^T\mathcal{X}$, i.e., $\mathcal{Y} \perp \mathcal{X}|\mathbf{W}^T\mathcal{X}$ [1].

Most of the SDR algorithms are developed based on the idea in [31, 32], which sets the SDR problem as an inverse regression problem. Some examples are sliced inverse regression (SIR) [31], principal Hessian directions (pHd) [32], sliced average variance estimation (SAVE), [13] and contour regression [30]. The main intuition behind these algorithms is to find the expectation $E(\mathcal{X}|\mathcal{Y})$, due to the fact that if the conditional distribution $P(\mathcal{Y}|\mathcal{X})$ varies along a subspace of $\mathcal{X}$, then the inverse regression $E(\mathcal{X}|\mathcal{Y})$ should also lie in $\mathcal{X}$ [31]. However, inverse regression always makes strong assumptions on the marginal distribution $P_{\mathcal{X}}(x)$ (e.g., the distribution should be elliptical), which is a limitation for dealing with non-elliptically distributed data.

In order to overcome the above problem, kernel dimensionality reduction (KDR) [19] was proposed as an alternative method. KDR makes no strong assumption about either the conditional distribution $P_{\mathcal{Y}|\mathbf{W}^T\mathcal{X}}(y|\mathbf{W}^T\mathcal{X})$ or the marginal distribution $P_{\mathcal{X}}(x)$. KDR measures the notion of conditional dependence by taking advantage of the conditional covariance operators defined on reproducing kernel Hilbert spaces. The conditional independence is then imposed by minimizing the conditional covariance operator in a RKHS.

[1] Here we use an abuse notation of $\mathbf{W}^T\mathcal{X}$ as the subspace after projection.

More specifically, for two RKHS mappings, $\mathbf{x} \to \boldsymbol{\phi}(\mathbf{x})$ and $\mathbf{y} \to \boldsymbol{\phi}(\mathbf{y})$ induced by universal kernels $k_{\mathbf{x}}(\cdot,\cdot)$ and $k_{\mathbf{y}}(\cdot,\cdot)$, respectively. We denote $E[\cdot]$ and $V(\cdot|\cdot)$ as expectation and conditional (co)variance operator, respectively. The following theorem holds [19]: $E[V(\boldsymbol{\phi}(\mathbf{x}))|\mathbf{x}] \preceq E[V(\boldsymbol{\phi}(\mathbf{x}))|\mathbf{z} = \mathbf{W}^T\mathbf{x}]$, where the equality holds if and only if $\mathbf{y} \perp \mathbf{x}|\mathbf{z} = \mathbf{W}^T\mathbf{x}$. This theorem asserts that any projection of the input generally increases uncertainty in predicting the response $\mathbf{y}$, with the minimal uncertainty achieved by the central subspace bases $\mathbf{W}$, i.e. $E[V(\boldsymbol{\phi}(\mathbf{x}))|\mathbf{x}]$. The idea behind KDR investigates that, $E[V(\boldsymbol{\phi}(\mathbf{x}))|\mathbf{z} = \mathbf{W}^T\mathbf{x}]$ can be expressed as the conditional covariance operator, $\boldsymbol{\Sigma}_{\mathbf{yy}|\mathbf{z}'}$ inducing $\mathbf{W}$ as the one that minimizes $\boldsymbol{\Sigma}_{\mathbf{yy}|\mathbf{z}}$, more precisely, its trace [20]

$$\begin{aligned} &\min_{\mathbf{W}} \operatorname{Tr}[\mathbf{K}_{\mathbf{y}}^c(\mathbf{K}_{\mathbf{z}}^c + n\epsilon\mathbf{I}_n)^{-1}] \\ &\text{s.t. } \mathbf{W}^T\mathbf{W} = \mathbf{I}, \end{aligned} \tag{2.13}$$

where $\mathbf{K}_{\mathbf{y}}^c$ and $\mathbf{K}_{\mathbf{z}}^c$ are centered kernel matrices computed by $\mathbf{K}_{\mathbf{y}}^c = \mathbf{H}\mathbf{K}_{\mathbf{y}}\mathbf{H}$ and $\mathbf{K}_{\mathbf{z}}^c = \mathbf{H}\mathbf{K}_{\mathbf{z}}\mathbf{H}$, respectively, and $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$.

The objective function of KDR is nonconvex, so the minimization of the objective function requires a nonlinear technique; Fuzumiku et.al [20] suggested to use the steepest descent method with line search. Moreover, to alleviate potential problems arising from local optima, a continuation method is used where the scale parameter (in Gaussian RBF kernel) is gradually decreased during the optimization process. Unfortunately, the computational burden is high due to the inversion of kernel matrix in each iterative step.

2.4 Comparison of PLS, SPCA and KDR

Based on the description of the three representative approaches for supervised dimensionality reduction, we now analyze the similarities and differences of the abovementioned PLS, SPCA and KDR.

Generally, all of the three methods try to find a faithful projection to capture the relationship of inputs and responses. PLS maximizes the *correlation* between the projected inputs and responses; SPCA maximizes the *dependence* between the projected inputs and responses based on HSIC; KDR minimizes the conditional covariance operator in RHKS to quantify the conditional dependence. As a consequence, PLS can only reveal the linear dependence between two blocks of variables, while SPCA shows stronger power for detecting a more general dependence, including both linear and nonlinear dependences. This observation can be demonstrated by their eigendecomposition solution.

$$\text{PLS} : (\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X})\mathbf{w} = \lambda\mathbf{w} \tag{2.14}$$

$$\text{SPCA} : (\mathbf{X}^T\mathbf{H}\mathbf{L}\mathbf{H}\mathbf{X})\mathbf{w} = \lambda\mathbf{w} \tag{2.15}$$

It can be seen clearly that when the kernel type of the output response in SPCA is selected as the linear kernel, SPCA and PLS are identical. However, the kernel for

response can be set as any other kind of kernel besides the linear one. This explains the reason why SPCA is superior to PLS on detecting dependence between two variables.

Differently, KDR formulates the supervised dimensionality reduction problem from the point of view of conditional independence. Statistically, KDR does not enforce strong assumptions on either the conditional distribution $P_{\mathcal{Y}|\mathbf{W}^T\mathcal{X}}(y|\mathbf{W}^T\mathcal{X})$ or the marginal distribution $P_{\mathcal{X}}(x)$, that offers more flexibility than the other methods of sufficient dimensionality reduction, e.g., SIR, pHd SAVE, and contour regression. The problem of KDR is its high computational cost because KDR cannot be solved in closed form as PLS and SPCA.

To demonstrate the performance of the three methods and analyze them in practice, here we implement them on two synthetic data sets. Assuming we are given n data points for training, $\{\mathbf{x}_i\}_{i=1}^n$ and each data point has features $\mathbf{x} = [X_1, \ldots, X_d]^T$.

2.4.1 Toy example one: plane

Firstly, we generate 1000 5-dimensional data points with Gaussian distribution, $\mathbf{x} \sim N(0, \mathbf{I}_5)$. Then, only the first two features are used for constructing the responses based on the following simple linear process,

$$y = 2X_1 + 3X_2 + \epsilon,$$

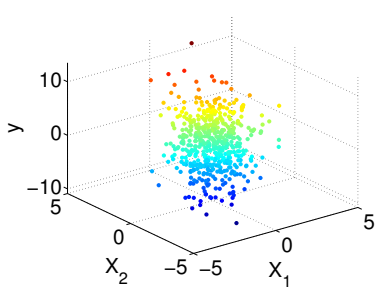
where the noise term $\epsilon \sim N(0, 0.5^2)$. Figure 2.1(a) shows the output responses of the test data with respect to two effective features X_1 and X_2, which is a plane. We use the first half of the points for training and the others for test. A projection matrix (vector) $\mathbf{W} \in \mathbb{R}^5$ is learned to project the data into one-dimensional space. The true projection matrix $\mathbf{W}$ is $[2, 3, 0, 0, 0]^T$, which only retains the first two informative features for regression.

For the test points, compared with the true projection in Figure 2.1(b), Figure 2.1(c-f) present the projection results of the three supervised methods, PLS, SPCA, KDR and the classical unsupervised approach PCA. In these figures, x-axis and y-axis represent the projected covariates and their true output responses, respectively. Each point in the plots is colored by its true response value y, where the value decreases along the color from red to blue. For this simple model based on linear function, all the three supervised methods find the correct projection directions. Because PCA does not consider the response information of data points, we cannot obtain informative projection results for regression.

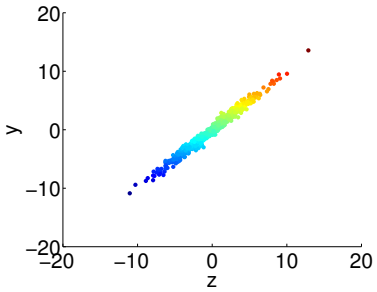
2.4.2 Toy example two: "parity"

Different from the first toy example, this time we will consider a regression problem with strong nonlinearity. This data set is generated by the "parity" function plus a noise term,

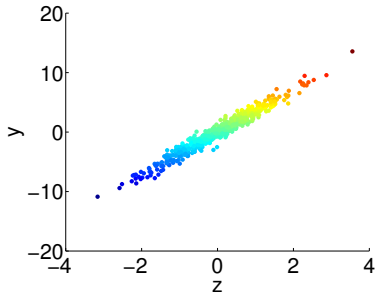
$$y = \sin(2\pi X_1)\sin(2\pi X_2) + \epsilon,$$



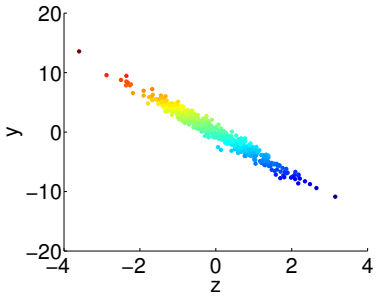
(a) $y = 2X_1 + 3X_2 + \epsilon$



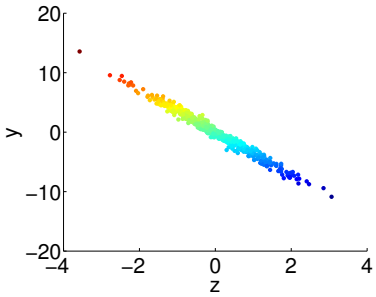
(b) true projection of test set



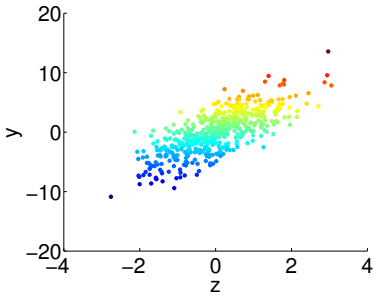
(c) PLS projection



(d) SPCA projection



(e) KDR projection

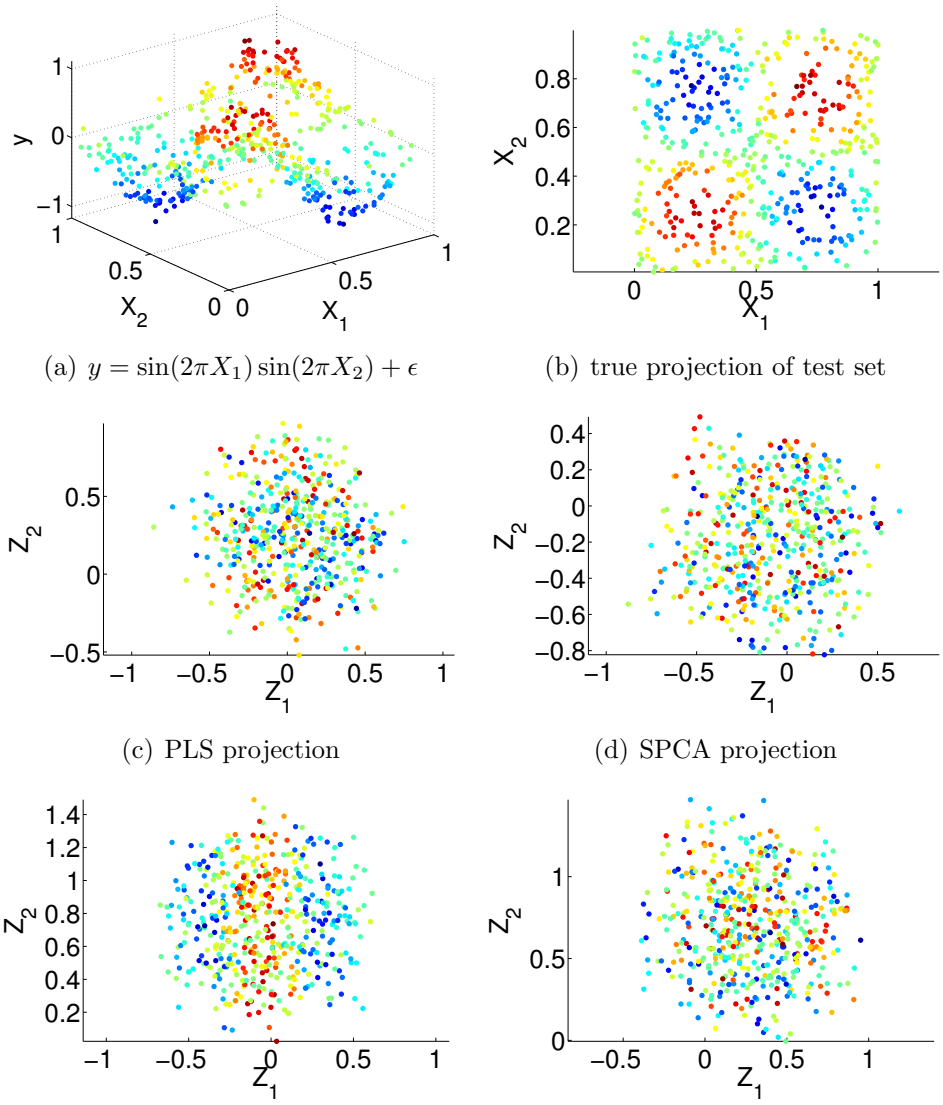


(f) PCA projection

Figure 2.1: Projection comparison on linear data.

where X_1 and X_2 are the first two features used for constructing the response, and the noise term $\epsilon \sim N(0, 0.1^2)$. We generate 1000 5-dimensional data points uniformly distributed in $[0, 1]^5$. The same splitting of the data set as toy example one, is used for training and test. Figure 2.2(a) shows the output response of "parity" function with respect to X_1 and X_2.

Clearly, after dimensionality reduction, only the first two features should be preserved as much as possible. The true two-dimensional projection is plotted in Figure 2.2(b). Figure 2.2(c-f) show the two-dimenional projection on test set by PLS, SPCA, KDR and PCA. As an unsupervised method, PCA again fails to obtain the effective projection for regression. Under such a nonlinear regression problem, the three supervised methods, PLS, SPCA and KDR are also unable to detect the first two informative features, which may not reveal the local structure of the data.



(a) $y = \sin(2\pi X_1)\sin(2\pi X_2) + \epsilon$

(b) true projection of test set

(c) PLS projection

(d) SPCA projection

(e) KDR projection

(f) PCA projection

Figure 2.2: Projection comparison on "parity" data.

To observe more details of the performances of the four methods, we list the learned projection matrix $\mathbf{W} \in \mathbb{R}^{5\times 2}$ below.

$$\mathbf{W}_{\text{PLS}} = \begin{bmatrix} 0.3356 & 0.3901 \\ 0.3299 & 0.6134 \\ 0.8082 & -0.5730 \\ -0.3396 & -0.3781 \\ 0.1000 & 0.0139 \end{bmatrix}, \mathbf{W}_{\text{SPCA}} = \begin{bmatrix} -0.2474 & -0.6743 \\ -0.3137 & -0.2945 \\ -0.7168 & -0.1486 \\ 0.3697 & -0.5390 \\ -0.4357 & 0.3820 \end{bmatrix},$$

$$\mathbf{W}_{\text{KDR}} = \begin{bmatrix} \underline{-0.7370} & -0.0514 \\ \underline{0.6753} & -0.0327 \\ \underline{-0.0091} & 0.8669 \\ \underline{-0.0247} & 0.2464 \\ \underline{-0.0042} & 0.4290 \end{bmatrix}, \mathbf{W}_{\text{PCA}} = \begin{bmatrix} -0.0838 & 0.6937 \\ 0.0075 & 0.0090 \\ 0.9469 & 0.2763 \\ -0.2052 & 0.3506 \\ -0.2328 & 0.5652 \end{bmatrix}.$$

We find that in both of two projection directions, PLS, SPCA and PCA give much weight to the three uninformative input variables, X_3, X_4 and X_5. For KDR, in the first projection direction, it identifies the first two useful features, X_1 and X_2, but in the second direction, X_3, X_4 and X_5 have the dominant weighting. Consequently, KDR still cannot provide a faithful projection into two-dimensional subspace. Another problem of KDR is its computational complexity. On a PC with 2.83GHz CPU and 4GB RAM, KDR spends much longer time for training, 65.77 seconds for 50 iterations compared with less than one second of training time for the other three methods.

From the theoretical and empirical comparison of the three approaches, we find that: for the simple linear data, all the three supervised DR algorithms can yield a faithful dimensionality reduction result; but for the data with strong nonlinearity, they are limited to find effective projection directions for regression, and for some of the algorithms, the computational complexity is unbearable for large-scale data sets. Thus, in this work, to overcome these drawbacks, we aim at designing a well-working algorithm for supervised dimensionality reduction, especially for regression. At the same time, the efficient optimization schemes should also be proposed.

Chapter 3

Supervised Distance Preserving Projection (SDPP)

The main problem of three methods in last chapter is that they only consider the global characteristics of data. Preserving local geometry plays a significant role in manifold learning algorithms, e.g., LLE, Laplacian eigenmap and ISOMAP, as mentioned in Section 1.1.1. These successful applications of preserving local geometry inspire us to explore its usefulness in supervised dimensionality reduction, especially for regression. Based on these considerations, in this chapter we propose a new criterion for supervised dimensionality reduction. This criterion is analyzed and cast into an optimization problem. Then, Different optimization schemes are implemented for an efficient computation. Finally, we evaluate our approach based on different parameter settings by a toy example.

3.1 Criterion

We start to motivate our approach by considering a continuous function $f : \mathcal{X} \mapsto \mathcal{Y}$. From Weierstrass definition, the continuity of the function f at point $\mathbf{x}' \in \mathcal{X}$ means that for every $\varepsilon_{\mathbf{x}} > 0$ there exists a $\varepsilon_y > 0$ such that for all $\mathbf{x} \in \mathcal{X}$:

$$\|\mathbf{x} - \mathbf{x}'\| < \varepsilon_{\mathbf{x}} \Rightarrow |f(\mathbf{x}) - f(\mathbf{x}')| < \varepsilon_y.$$

This definition of continuity claims that if two input covariates $\mathbf{x}$ and $\mathbf{x}'$ are close to each other, their responses $y = f(\mathbf{x})$ and $y' = f(\mathbf{x}')$ should also be close. Since we focus on the task of dimensionality reduction for regression now, we try to reduce the dimensionality without loss of the continuity of the original regression function f. To preserve the continuity, we resort to considering the local geometry of the input space $\mathcal{X}$ and response $\mathcal{Y}$. After dimensionality reduction, we aim to minimize the difference between the *l*ocal dissimilarities in the $\mathcal{X}$-space and $\mathcal{Y}$-space.

Here we assume the subspace $\mathcal{Z}$ of $\mathcal{X}$ is obtained by a linear transformation, that is, for the input $\mathbf{x}$, the new representation in the subspace is $\mathbf{z} = \mathbf{W}^T\mathbf{x}$, where the projection matrix $\mathbf{W} \in \mathbb{R}^{d\times r}$. Concretely, we seek the linear transformation by

minimizing the following criterion,

$$\mathcal{J}(\mathbf{W}) = \frac{1}{n}\sum_{i}\sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)} (d_{ij}^2 - \delta_{ij}^2)^2, \tag{3.1}$$

where $\mathcal{N}(\mathbf{x}_i)$ is a neighborhood of $\mathbf{x}_i$, $d_{ij}^2 = \|\mathbf{z}_i - \mathbf{z}_j\|^2$ is the square of Euclidean distance between $\mathbf{z}_i$ and $\mathbf{z}_j$ and $\delta_{ij}^2 = \|y_i - y_j\|^2$ is the square of Euclidean distance between y_i and y_j.

Alternatively, the criterion of SDPP can also be written in a more compact form,

$$\mathcal{J}(\mathbf{W}) = \frac{1}{n}\sum_{ij} \mathbf{G}_{ij}(\mathbf{D}_{ij} - \mathbf{\Delta}_{ij})^2, \tag{3.2}$$

where $\mathbf{D}_{ij} = d_{ij}^2$, $\mathbf{\Delta}_{ij} = \delta_{ij}^2$ and neighborhood graph $\mathbf{G}$ is defined as follows,

$$\mathbf{G}_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_j \text{ is a neighbor of } \mathbf{x}_i, \\ 0 & \text{otherwise.} \end{cases}$$

We refer to this novel criterion for dimensionality reduction as Supervised Distance Preserving Projection (SDPP). We observe that, to achieve the continuity preservation during DR, SDPP attempts to *match* the local geometrical structure between the subspace $\mathcal{Z}$ and response space $\mathcal{Y}$ as much as possible. Here, the local geometrical structure is expressed by the pairwise distances over the neighborhoods of the input covariates. This interpretation of SDPP can be illustrated in Figure 3.1. From this point of view, SDPP retains the information of original function between the inputs and response as much as possible, which is beneficial for the regression in the subspace.

Despite the simple formulation, SDPP has several advantages for DR in regression tasks:

- SDPP is possible to work for the cases with multiple responses because our model is based on the pairwise distances, where the Euclidean distance for two response vectors is still valid.
- Since SDPP learns an affine transformation matrix, this parametric mapping can easily find the projection of out-of-sample data.

3.2 Optimization of SDPP

For the minimization of the objective function in Equation 3.1, we have

$$d_{ij}^2 = (\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{W}\mathbf{W}^T(\mathbf{x}_i - \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{P}(\mathbf{x}_i - \mathbf{x}_j),$$

where $\mathbf{P} = \mathbf{W}\mathbf{W}^T$ is a positive semidefinite (PSD) matrix, i.e., $\mathbf{P} \succeq 0$. For notational simplicity, we denote $\boldsymbol{\tau}_{ij} = \mathbf{x}_i - \mathbf{x}_j$, thus,

$$d_{ij}^2 = \boldsymbol{\tau}_{ij}^T\mathbf{P}\boldsymbol{\tau}_{ij} = vec(\boldsymbol{\tau}_{ij}\boldsymbol{\tau}_{ij}^T)^T vec(\mathbf{P}) = \mathbf{l}_{ij}^T\mathbf{p}, \tag{3.3}$$

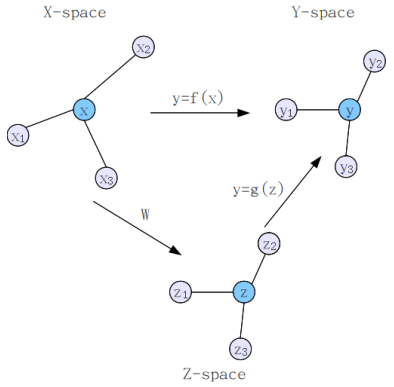


Figure 3.1: Illustration of SDPP

where the vector $\mathbf{l}_{ij} = vec(\boldsymbol{\tau}_{ij}\boldsymbol{\tau}_{ij}^T), \mathbf{p} = vec(\mathbf{P})$, and $vec(\cdot)$ is an operator that concatenates all the columns of a matrix into a new vector.

Now we formulate the optimization problem into an instance of *Convex Quadratic Semidefinite Programming* (QSDP) [7]. The objective function can be rewritten as

$$\begin{aligned}\mathcal{J}(\mathbf{p}) &= \mathbf{p}^T(\underbrace{\frac{1}{n}\sum_i \sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)} \mathbf{l}_{ij}\mathbf{l}_{ij}^T}_{\mathbf{A}})\mathbf{p} + (\underbrace{-\frac{2}{n}\sum_i \sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)} \delta_{ij}^2 \mathbf{l}_{ij}}_{\mathbf{b}})^T\mathbf{p}_v \\ &+ \underbrace{\frac{1}{n}\sum_i \sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)} \delta_{ij}^4}_{c} \\ &= \mathbf{p}^T\mathbf{A}\mathbf{p} + \mathbf{b}^T\mathbf{p} + c, \end{aligned} \tag{3.4}$$

where $\mathbf{A}$ is symmetric of the size $d^2 \times d^2, \mathbf{b} \in \mathbb{R}^{d^2}$, and c is a constant that can be ignored later in optimization. Then, SDPP optimization becomes the following QSDP problem,

$$\begin{aligned} &\min_{\mathbf{p}} \ \mathbf{p}^T\mathbf{A}\mathbf{p} + \mathbf{b}^T\mathbf{p} \\ &\text{s.t. } \mathbf{P} \succeq 0 \end{aligned} \tag{3.5}$$

Note that the objective of the QSDP problem in Equation 3.5 is a quadratic function with respect to the matrix $\mathbf{P}$. The similar QSDP formulations also arise in colored MVU [49], Conformal Eigenmaps [47], Kernel Matrix Completion [22]. Here we will introduce two kinds of existing methods for addressing the QSDP.

3.2.1 Semidefinite Programming Formulation for SDPP

Before we formulate the QSDP in Equation 3.5 into the Semidefinite Programming (SDP) problem, it is necessary to have a brief overview on SDP.

Semidefinite Programming (SDP)

In SDP, one minimizes a linear function subject to the constraint that an affine combination of symmetric matrices is positive semidefinite [55]. Such a constraint is nonlinear and nonsmooth, but convex, so semidefinite programs are convex optimization problems [9]. Semidefinite programming unifies several standard problems (e.g., linear and quadratic programming) and finds many applications in engineering and combinatorial optimization. The following is an example of SDP problem:

$$\begin{aligned} \min_{\mathbf{X}} \ & \mathbf{X}_{13} \\ \text{s.t.} \ & -0.2 \leq \mathbf{X}_{12} \leq -0.1 \\ & 0.4 \leq \mathbf{X}_{23} \leq 0.5 \\ & \mathbf{X}_{11} = \mathbf{X}_{22} = \mathbf{X}_{33} = 1 \\ & \mathbf{X} \succeq 0, \end{aligned}$$

where $\mathbf{X}$ is a 3×3 matrix and the semidefinite constraint is $\mathbf{X} \succeq 0$.

Although semidefinite programs are much more general than the linear programs, they are not much harder to solve. Most interior-point methods for linear programming have been generalized to semidefinite programs. There are many toolboxes available for solving SDP, for example, SeDuMi [50] and YALMIP [35].

Schur complement

In linear algebra and the theory of matrices, the Schur complement of a matrix block (i.e., a submatrix within a larger matrix) is defined as follows. Suppose $\mathbf{C}_1$, $\mathbf{C}_2$, $\mathbf{C}_3$, $\mathbf{C}_4$ are respectively $p \times p$, $p \times q$, $q \times p$ and $q \times q$ matrices, and $\mathbf{C}_1$ is invertible. Let

$$\mathbf{M} = \begin{bmatrix} \mathbf{C}_1 & \mathbf{C}_2 \\ \mathbf{C}_3 & \mathbf{C}_4 \end{bmatrix},$$

where $\mathbf{M}$ is with the size $(p+q) \times (p+q)$. Then the Schur complement of the block $\mathbf{C}_1$ of the matrix $\mathbf{M}$ is the $q \times q$ matrix

$$\mathbf{S}_c = \mathbf{C}_4 - \mathbf{C}_3 \mathbf{C}_1^{-1} \mathbf{C}_2.$$

Now we consider a characterization of symmetric positive semidefinite matrices using Schur complements. Assume the matrix $\mathbf{M}$ is symmetric of the form

$$\mathbf{M} = \begin{bmatrix} \mathbf{C}_1 & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C}_4 \end{bmatrix},$$

if $C_1 \succeq 0$, then the following property hold: $\mathbf{M} \succeq 0$ if and only if the Schur complement $\mathbf{S}_c = \mathbf{C}_4 - \mathbf{B}^T \mathbf{C}_1^{-1} \mathbf{B} \succeq 0$ [64].

Based on this characterization of symmetric PSD matrices using Schur complements, now we rewrite the QSDP problem into a SDP problem. Firstly, we introduce a slack variable μ acting as an upper bound of $\mathbf{p}^T \mathbf{A} \mathbf{p}$ in Equation 3.5,

$$\mu \geq \mathbf{p}^T \mathbf{A} \mathbf{p}.$$

Then we do the factorization $\mathbf{p}^T\mathbf{A}\mathbf{p} = (\mathbf{A}^{\frac{1}{2}}\mathbf{p})^T(\mathbf{A}^{\frac{1}{2}}\mathbf{p})$, where $\mathbf{A}^{\frac{1}{2}}$ is the matrix square root of $\mathbf{A}$. The following inequality holds,

$$\mu - (\mathbf{A}^{\frac{1}{2}}\mathbf{p})^T\mathbf{I}_{d^2}^{-1}(\mathbf{A}^{\frac{1}{2}}\mathbf{p}) \geq 0,$$

where $\mathbf{I}_{d^2}$ is the $d^2 \times d^2$ identity matrix. We find that $\mu - (\mathbf{A}^{\frac{1}{2}}\mathbf{p})^T\mathbf{I}_{d^2}^{-1}(\mathbf{A}^{\frac{1}{2}}\mathbf{p})$ is the Schur complement of $\mathbf{I}_{d^2}$ in the large matrix

$$\mathbf{M}_{\text{aux}} = \begin{bmatrix} \mathbf{I}_{d^2} & \mathbf{A}^{\frac{1}{2}}\mathbf{p} \\ (\mathbf{A}^{\frac{1}{2}}\mathbf{p})^T & \mu \end{bmatrix}.$$

Due to the positive semidefiniteness of the matrix $\mathbf{I}_{d^2}$ and its Schur complement, the large matrix $\mathbf{M}_{\text{aux}}$ is also positive semidefinite.

According to the derivation above, we have an equivalent SDP formulation for the QSDP presenting as follows:

$$\begin{aligned} &\min_{\mathbf{p},\mu} \mu + \mathbf{b}^T\mathbf{p} \\ &\text{s.t. } \mathbf{P} \succeq 0 \\ &\qquad \begin{bmatrix} \mathbf{I}_{d^2} & \mathbf{A}^{\frac{1}{2}}\mathbf{p} \\ (\mathbf{A}^{\frac{1}{2}}\mathbf{p})^T & \mu \end{bmatrix} \succeq 0. \end{aligned} \tag{3.6}$$

The similar reformulation of the QSDP problem also exist in [47, 28, 33].

We observe that this SDP formulation for the QSDP has two semidefinite cone constraints, with size $d \times d$ and $(d^2+1) \times (d^2+1)$, respectively. They scale poorly and causes difficulties to process numerically [7]. Additionally, in the dual form of SDP in Equation 3.6, the number of iterations required by classical interior-point algorithms is $O(d)$, and the total number of arithmetic operations needed is $O(d^9)$ [7]. In the work [61], the experiments for the SDP problem were conducted in Matlab 7.6.0(R2008a) on a PC with 2.5GHz CPU and 4GB RAM. The reported results show that when the dimension d is very small, e.g., $d = 10$, it spends only a few seconds to solve this SDP problem, but takes more than one day when $d = 40$.

3.2.2 SQLP Formulation for SDPP

In this thesis, we apply the idea from [61], which takes advantage of low rank structure of $\mathbf{A}$ and reformulates the QSDP into the *Semidefinite Quadratic Linear Programming* (SQLP) problem. Concretely, since matrix $\mathbf{A}$ is symmetric positive semidefinite, we can factorize $\mathbf{A}$ as $\mathbf{A} = \mathbf{B}^T\mathbf{B}$ with Cholesky factorization, where $\mathbf{B} \in \mathbb{R}^{q \times d^2}$ and q is the rank of $\mathbf{B}$. Assume $\mathbf{f} = \mathbf{B}\mathbf{p}$, then the QSDP problem (3.5) can be written as

$$\min_{\mathbf{p},\mathbf{f},\mu} \mu + \mathbf{b}^T\mathbf{p} \tag{3.7}$$

$$\text{s.t. } \mathbf{f} = \mathbf{B}\mathbf{p}, \tag{3.8}$$

$$\mathbf{f}^T\mathbf{f} \leq \mu, \tag{3.9}$$

$$\mathbf{P} \succeq 0. \tag{3.10}$$

In work [61], the authors show that the constraint in (3.9) is equivalent to a second-order cone constraint. We define $\mathbb{K}_m$ as the m-dimensional second-order cone constraint,

$$\mathbb{K}_m = \{[x_0; \mathbf{x}] \in \mathbb{R}^m | x_0 \geq \|\mathbf{x}\|\}.$$

Now we introduce an auxiliary vector $\mathbf{u} = [\frac{1+\mu}{2}, \frac{1-\mu}{2}, \mathbf{f}^T]^T$, then following theorem holds (see [61] for proof): $\mathbf{f}^T\mathbf{f} \leq \mu$ if and only if $\mathbf{u} \in \mathbb{K}_{q+2}$. Let $\mathbf{e}_i$ be the ith basis vector with length of $q+2$, and $\mathbf{C} = [\mathbf{0}_{q\times 2}, \mathbf{I}_{q\times q}]$. Thus, we have following equalities,

$$\begin{aligned}
&(\mathbf{e}_1 - \mathbf{e}_2)^T\mathbf{u} = \mu, \\
&(\mathbf{e}_1 + \mathbf{e}_2)^T\mathbf{u} = 1, \\
&\mathbf{f} = \mathbf{C}\mathbf{u}.
\end{aligned}$$

Based on these manipulations, the problem (3.7-3.10) can be reformulated as:

$$\begin{aligned}
\min_{\mathbf{p},\mathbf{u}} \ & (\mathbf{e}_1 - \mathbf{e}_2)^T\mathbf{u} + \mathbf{b}^T\mathbf{p} \\
\text{s.t. } & (\mathbf{e}_1 + \mathbf{e}_2)^T\mathbf{u} = 1, \\
& \mathbf{B}\mathbf{p} - \mathbf{C}\mathbf{u} = 0, \\
& \mathbf{u} \in \mathbb{K}_{q+2}, \\
& \mathbf{P} \succeq 0,
\end{aligned} \tag{3.11}$$

which is an instance of SQLP problem and be solved efficiently.

In this reformulation, the semidefinite cone constraint of size $(d^2+1) \times (d^2+1)$ in Equation 3.6 is replaced with one second-order cone constraint of size $(q+2)$ and $(q+1)$ linear constraints, which are much easier to optimize numerically and can be solved more efficiently. Thanks to utilizing the low rank structure of $\mathbf{A}$, when we use standard interior-point algorithm, the number of arithmetic operations required is $O(\sqrt{d})$ compared with $O(d)$ in SDP formulation. Additionally, the total number of arithmetic operations required is $O(d^{6.5})$ compared with $O(d^9)$ in the SDP form. The Matlab toolbox SDPT3-4.0 [53, 54] can be used to solve the SQLP in practice.

After learning the matrix $\mathbf{P}$, we can factorize $\mathbf{P}$ as $\mathbf{P} = \mathbf{W}\mathbf{W}^T$. One reasonable way is to apply Singular Value Decomposition (SVD) on $\mathbf{P}$, and then the ith column of $\mathbf{W}$ is $\sqrt{\lambda_i}\mathbf{v}_i$, where λ_i and $\mathbf{v}_i$ are the ith eigenvalue and eigenvector of $\mathbf{P}$, respectively.

Determining Dimension r of the Subspace

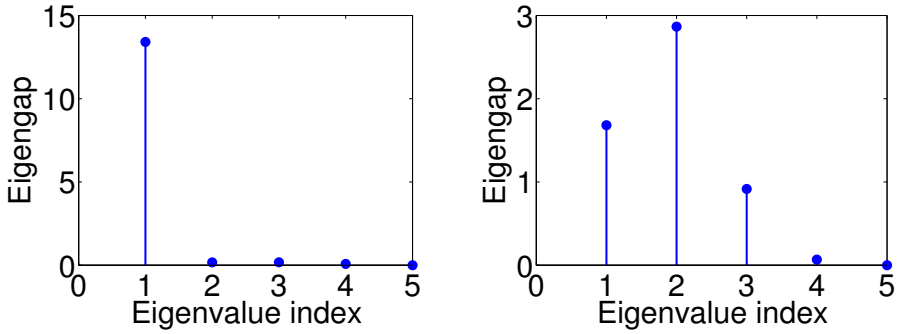
Generally, the dimension of the subspace is given beforehand by user or determined by cross-validation according to a specific prediction model. Fortunately, in the SQLP optimization of SDPP, the low-rank structure of $\mathbf{P}$ benefits us to determine the dimension of the subspace. We take advantage of so-called "eigengap" [56], i.e., the gap between eigenvalues in SVD. Assume the eigenvalues of $\mathbf{P}$ are sorted by descending order, $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$, the eigengap is defined as

$$\gamma_i = \lambda_i - \lambda_{i+1}, \ i = 1, \ldots, d-1.$$

We then choose the index of eigenvalue with the maximal eigengap as the optimal dimension for the projection subspace,

$$r_{\text{opt}} = \arg\max_i \gamma_i.$$

In our further experiments when applying the SQLP formulation to solve SDPP, this heuristic is used to identify the optimal dimension for the projection space. We plot the eigengap in Figure 3.2 for the two synthetic data sets linear and "parity" data, where the true dimension is 1 and 2, respectively. We can see that the eigengap successfully identifies the true dimensions of the subspaces.



(a) Linear data (b) "Parity" data

Figure 3.2: Eigengap plots for linear, "parity"and curve data sets.

3.2.3 Conjugate-Gradient Optimization for SDPP

The above-mentioned two optimization methods the SDP and the SQLP for SDPP do not optimize the projection matrix $\mathbf{W}$ directly, instead, both of them optimize the PSD matrix $\mathbf{P} = \mathbf{W}\mathbf{W}^T$. The advantage of these formulations is that they set the SDPP problem into the standard framework of optimization with semidefinite or second-order cone constraint, that can be solved by many popular optimization libraries or toolboxes, e.g., SeDuMi [50] and SDPT3-4.0 [53, 54]. When the dimensionality $d < 100$, these toolboxes work efficiently for the SQLP problem based on our experimental validation. However, the parameter matrix $\mathbf{A}$ is with the size $d^2 \times d^2$. When the dimensionality of original data set is high, $\mathbf{A}$ becomes extremely large and brings impossibility for storage and further optimization even shielded by efficient optimization toolboxes. Regarding to the poor performance of the SDP and SQLP formulations under high-dimensional data, it is necessary to develop an optimization algorithm that scales well with respect to the dimensionality. In this thesis, the Conjugate-Gradient (CG) optimization approach is used for handling this problem.

CG optimization method works when the cost function is twice differentiable at the minimum, which is satisfied by the objective function of SDPP. The CG algorithm selects the successive direction vectors as a conjugate version of the successive

gradients obtained as the method progresses. Thus, the directions are not specified beforehand, but rather are determined sequentially at each step of the iteration. At step t one evaluates the current negative gradient vector and adds to it a linear combination of the previous direction vectors to obtain a new conjugate direction vector along which to move.

In our case, the objective function and its gradient of SDPP are listed below:

$$\mathcal{J}(\mathbf{W}) = \frac{1}{n}\sum_{ij}\mathbf{G}_{ij}(\mathbf{D}_{ij} - \mathbf{\Delta}_{ij})^2, \tag{3.12}$$

$$\nabla_{\mathbf{W}}\mathcal{J} = \frac{4}{n}\sum_{ij}\mathbf{G}_{ij}(\mathbf{D}_{ij} - \mathbf{\Delta}_{ij})\boldsymbol{\tau}_{ij}\boldsymbol{\tau}_{ij}^T\mathbf{W}. \tag{3.13}$$

Denote $\mathbf{Q} = \mathbf{G}\odot(\mathbf{D}-\mathbf{\Delta})$, where $\odot$ represents the element-wise product of two matrices, symmetric matrix $\mathbf{R} = \mathbf{Q}+\mathbf{Q}^T$, $\mathbf{S}$ a diagonal matrix with $\mathbf{S}_{ii} = \sum_j \mathbf{R}_{ij}$. Following some algebraic manipulations, we obtain a more compact form of the gradient $\nabla_{\mathbf{W}}\mathcal{J}$,

$$\begin{aligned}
\nabla_{\mathbf{W}}\mathcal{J} &= \frac{4}{n}\sum_{ij}\mathbf{Q}_{ij}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{W} \\
&= \frac{4}{n}\sum_{ij}(\mathbf{x}_i\mathbf{Q}_{ij}\mathbf{x}_i^T + \mathbf{x}_j\mathbf{Q}_{ij}\mathbf{x}_j^T - \mathbf{x}_i\mathbf{Q}_{ij}\mathbf{x}_j^T - \mathbf{x}_j\mathbf{Q}_{ij}\mathbf{x}_i^T)\mathbf{W} \\
&= \frac{4}{n}\left[\sum_{ij}\mathbf{x}_i(\mathbf{Q}_{ij} + \mathbf{Q}_{ji})\mathbf{x}_i^T - \sum_{ij}\mathbf{x}_i(\mathbf{Q}_{ij} + \mathbf{Q}_{ji})\mathbf{x}_j^T\right]\mathbf{W} \\
&= \frac{4}{n}\left(\sum_{ij}\mathbf{x}_i\mathbf{R}_{ij}\mathbf{x}_i^T - \sum_{ij}\mathbf{x}_i\mathbf{R}_{ij}\mathbf{x}_j^T\right)\mathbf{W} \\
&= \frac{4}{n}\left(\sum_{i}\mathbf{x}_i\sum_{j}\mathbf{R}_{ij}\mathbf{x}_j^T - \sum_{ij}\mathbf{x}_i\mathbf{R}_{ij}\mathbf{x}_j^T\right)\mathbf{W} \\
&= \frac{4}{n}\left(\sum_{i}\mathbf{x}_i\mathbf{S}_{ii}\mathbf{x}_j^T - \sum_{ij}\mathbf{x}_i\mathbf{R}_{ij}\mathbf{x}_j^T\right)\mathbf{W} \\
&= \frac{4}{n}\left(\mathbf{X}^T\mathbf{S}\mathbf{X} - \mathbf{X}^T\mathbf{R}\mathbf{X}\right)\mathbf{W} \\
&= \frac{4}{n}\mathbf{X}^T(\mathbf{S} - \mathbf{R})\mathbf{X}\mathbf{W},
\end{aligned} \tag{3.14}$$

where each row of data matrix $\mathbf{X}$ is a data point $\mathbf{x}_i$ and $\mathbf{S} - \mathbf{R}$ is the well-known Laplacian matrix [11]. The form of gradient in Equation 3.14 is much more compact and easier to calculate than that in Equation 3.13. We summarize the CG optimization for SDPP in Algorithm 3.

Since the CG directly optimizes the projection matrix $\mathbf{W}$, the dimension r of the subspace must be given before optimization or determined by cross-validation according to the specific prediction model.

Algorithm 3 Conjugate-Gradient optimization for SDPP

Input: training data matrix $\mathbf{X}$ and $\mathbf{Y}$, neighborhood graph $\mathbf{G}$, initialized projection matrix $\mathbf{W}_0$

Output: optimized projection matrix $\mathbf{W}$.

1. Compute gradient $\nabla_{\mathbf{W}}\mathcal{J}$;
2. Vectorize the projection matrix, $\mathbf{w}_0 = vec(\mathbf{W}_0)$;
3. Vectorize the gradient, $\mathbf{g}_0 = vec(\nabla_{\mathbf{W}}\mathcal{J})$;
4. Initialize the conjugate direction as $\Lambda\mathbf{w}_0 = -\mathbf{g}_0$;
for $t = 1 \rightarrow T$ **do**
  5. Calculate β_t by Polak-Ribiére's rule, $\beta_t = \frac{\mathbf{g}_t^T(\mathbf{g}_t - \mathbf{g}_{t-1})}{\mathbf{g}_{t-1}^T\mathbf{g}_{t-1}}$;
  6. Update the conjugate direction, $\Lambda\mathbf{w}_t = -\mathbf{g}_t + \beta_t\Lambda\mathbf{w}_{t-1}$;
  7. Perform line search, $\eta_t = \arg\min_\eta \ \mathcal{J}(\mathbf{w} + \eta\Lambda\mathbf{w}_t)$;
  8. Update $\mathbf{w}$, $\mathbf{w}_{t+1} = \mathbf{w}_t + \eta_t\Lambda\mathbf{w}_t$
end for
9. Reshape the vector $\mathbf{w}_{T+1}$ into the matrix $\mathbf{W}$, which is the optimized projection matrix.

3.3 Neighborhood Selection Strategies for SDPP

In the criterion of SDPP, for each training point, we need to determine its neighbors and construct neighborhood graph for further computation and optimization. In this section, we will describe several neighborhood selection strategies for SDPP, from a geometrical point of view.

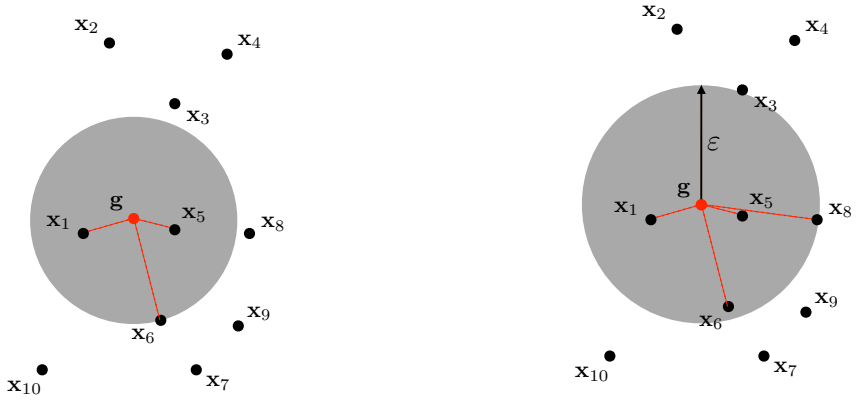
We divide the different neighborhoods into two categories: *not-enclosing neighborhood* and *enclosing neighborhood*, based on whether the neighborhood $\mathcal{N}(\mathbf{g})$ always encloses the current training point $\mathbf{g}$. If $\mathcal{N}(\mathbf{g})$ always encloses $\mathbf{g}$, we call it an enclosing neighborhood; that is, $\mathbf{g} \in \text{conv}(\mathcal{N}(\mathbf{g}))$, where the convex hull of a point set $S=\{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ is defined as $\text{conv}(S)=\{\sum_{i=1}^n \omega_i\mathbf{s}_i | \sum_{i=1}^n \omega_i = 1, \omega_i \geq 0\}$. Intuitively, the enclosing neighborhoods can provide geometrically balanced neighbors. Now, we overview the two kinds of neighborhood selection strategies and explore their performances in SDPP by a toy example.

3.3.1 Not-enclosing neighborhoods

Firstly, we overview two different strategies for defining a not-enclosing neighborhood: classic k-nearest neighborhood (kNN) and ε-neighborhood.

k-nearest neighborhood (kNN)

In the area of machine learning, kNN is the most common-used neighborhood construction strategies. Classic kNN defines a neighborhood of $\mathbf{g}$ using the first k of its nearest neighbors, according to a specified distance metric. Usually, the Euclidean metric is used and the number of neighbors k is fixed or cross-validated. Figure 3.3(a) shows an example of a kNN neighborhood of size $k = 3$ for the point $\mathbf{g}$, where its neighborhood $\mathcal{N}(\mathbf{g})^{kNN}=\{\mathbf{x}_1, \mathbf{x}_5, \mathbf{x}_6\}$.



(a) k-nearest neighborhood (b) ε-neighborhood

Figure 3.3: Not-enclosing neighborhoods.

The ε-neighborhood

For each training point $\mathbf{g}$, we treat the point $\mathbf{x}_i$ as its neighbor if the distance between them is smaller than ε. Figure 3.3(b) is an example of a ε-neighborhood for $\mathbf{g}$, where its ε-neighborhood: $\mathcal{N}(\mathbf{g})^{\varepsilon} = \{\mathbf{x}_1, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_8\}$.

According to the definitions of these two neighborhoods, we have the following important observations. In the ε-neighborhood, when we have data distributing in different regions (sparse or dense) of the space, i.e., in different regions of the space, the distances between points are different, we cannot find any neighbors for some points distributing in the sparse regions with small ε. In this case, the parameter ε is often difficult to determine in practice. A commonly-used heuristic for selecting ε is $(\log(n)/n)^{1/d}$ [40]. On the other hand, k-nearest neighborhood avoids this problem to guarantee that each point has k neighbors, that is preferred in our algorithm SDPP.

However, for kNN, the determination of the parameter k is still an open problem in many machine learning problems. Currently, barely any theoretical results are known to guide us in this task. There are only some heuristics for selecting k. In our algorithm, we use the heuristic that chooses k in the order of $\log(n)$, which is popularly used in spectral clustering [10, 56]. For the experiments conducted in this thesis, we find this heuristic works effectively for SDPP.

3.3.2 Enclosing neighborhoods

Enclosing k-nearest neighbors (ekNN):

It is based on the kNN of the point $\mathbf{g}$ and extends it to define a neighborhood that encloses it. ekNN is the neighborhood of the nearest neighbors with the smallest k

such that $\mathbf{g} \in \text{conv}(\mathcal{N}(\mathbf{g}, k))$, where $\mathcal{N}(\mathbf{g}, k)$ is the set of kNNs of $\mathbf{g}$ [25]. If $\mathbf{g}$ is outside of convex hull of the set $\mathcal{X} \backslash \mathbf{g}$, no such k exists. Define *distance to enclosure* as

$$D(\mathbf{g}, \mathcal{N}(\mathbf{g})) = \min_{\mathbf{s} \in \text{conv}(\mathcal{N}(\mathbf{g}))} \|\mathbf{g} - \mathbf{s}\|_2, \tag{3.15}$$

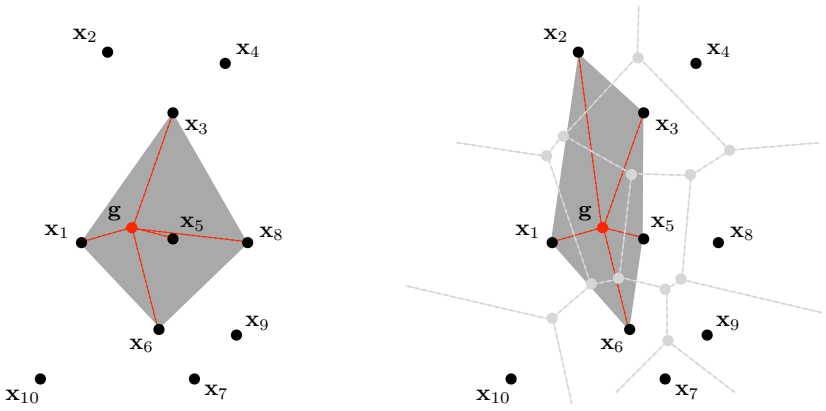
where $\mathbf{s}$ is any point in the convex hull around the neighborhood of $\mathbf{g}$. Note that $D(\mathbf{g}, \mathcal{N}(\mathbf{g})) = 0$ only if $\mathbf{g} \in \text{conv}(\mathcal{N}(\mathbf{g}))$. Then, the e$k$NN neighborhood is $\mathcal{N}(\mathbf{g}, k^*)$ with

$$k^* = \min_k \{k | D(\mathbf{g}, \mathcal{J}_\mathbf{g}(k)) = D(\mathbf{g}, \mathcal{X})\}. \tag{3.16}$$

The computational complexity for building a convex hull using k neighbors is $(k^{\lfloor d/2 \rfloor})$, where $\lfloor \cdot \rfloor$ is a floor function. For large-scale and very high-dimensional data, the computational cost of ekNN is high because for each training point, we need to increase the number of the nearest neighbors and recompute the convex hull. An example of ekNN is presented in Figure 3.4(a), where $\mathcal{N}(\mathbf{g})^{ekNN} = \{\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_8\}$.

Natural neighbors (NN)

Natural neighbors are based on the Voronoi tessellation of the training points [48]. The natural neighbors of $\mathbf{g}$ are defined as those points whose Voronoi cells are adjacent to the cell including $\mathbf{g}$. The computational complexity of Voronoi tessellation is related to both the number of points and dimensions: $O(n \log(n))$ when $d < 3$ and $O((n/d)^{d/2})$ when $d \geq 3$. Clearly, the Voronoi tessellation is much computationally involving in very high-dimensional cases. Figure 3.4(b) shows an example of natural neighbors, where $\mathcal{N}(\mathbf{g})^{NN} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_5, \mathbf{x}_6\}$.



(a) Enclosing k-nearest neighborhood (b) Natural neighborhood

Figure 3.4: Enclosing neighborhoods.

Although the two enclosing neighborhoods, ekNN and NN are both parameter-free and geometrically balanced, their computational complexities increase exponentially with respect to dimensionality and size of the data set. When handling large-scale and high-dimensional data, the computational costs of ekNN and NN are unbearable.

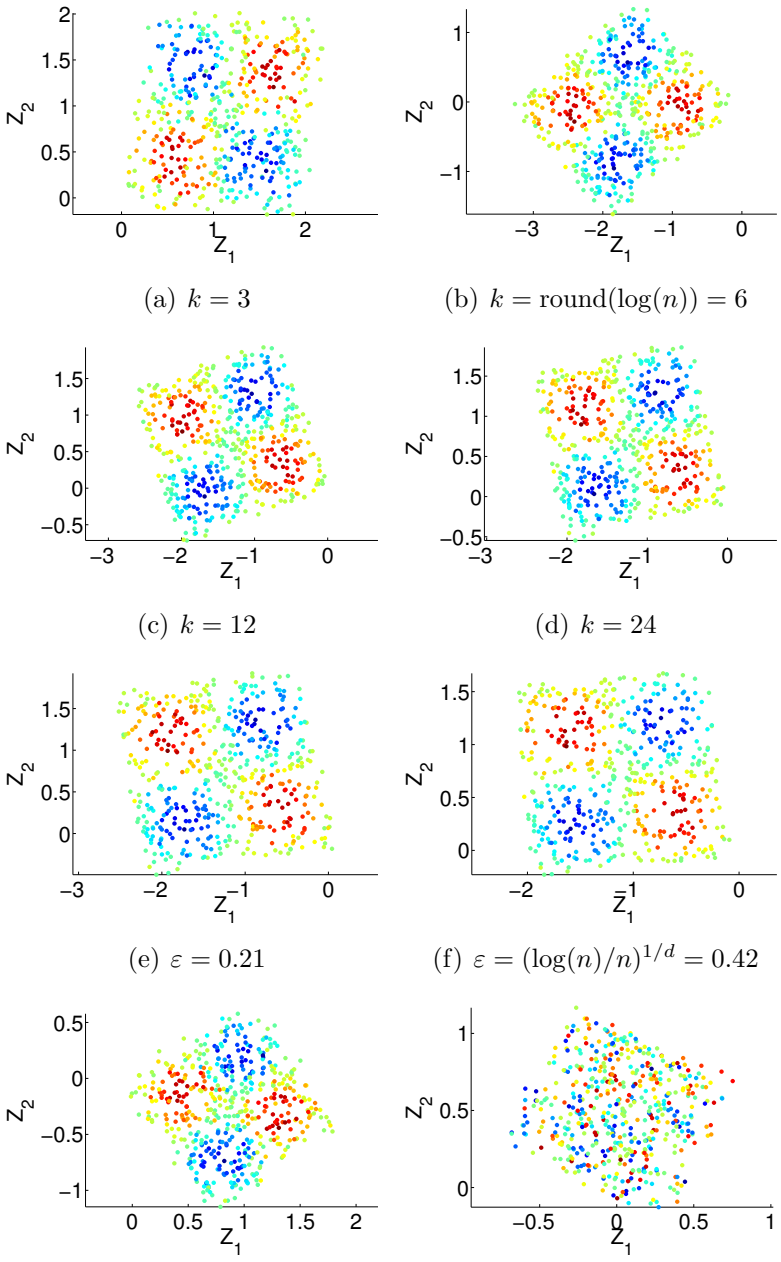
3.3.3 Evaluation by A Toy Example

Now we evaluate the performance of SDPP under the two types of neighborhoods. The synthetic data for the evaluation is the "parity" data in Section 2.4. The same experimental settings are used, including the process of data generation and splitting of the training and test sets.

For the not-enclosing neighborhoods, Figure 3.5 shows the two-dimensional projection results on the test sets. In the settings of kNN, around the heuristic $k = \text{round}(\log(n)) = 6$, we use the half, 2 times and 4 times of this heuristic to compare their performances in SDPP. The operator round$(\cdot)$ rounds a real number to its nearest integer. We observe that when $k = 6$, 12 and 24 in Figure 3.5(b-d), faithful projections for visualization are obtained, and the four centers indicating the two peaks and two valleys are more clear than that when using less nearest neighbors. For the "parity" data, kNN neighborhood is robust to the number of the nearest neighbors in the range $k \in [6, 24]$. For the ε-neighborhood, the same strategy as kNN is used to set the values of ε for comparison. However, from the projections results in Figure 3.5(e-h), we can see that the performance of SDPP is empirically sensitive to the value of ε. A large ε can totally ruin the correct projection structure, as shown in Figure 3.5(g) and Figure 3.5(h).

For the enclosing neighborhoods, the projection results based on ekNN and NN are presented in Figure 3.6. Enclosing kNN obtains a better results than natural neighborhood, but worse than the results from kNN. Additionally, compared with not-enclosing neighborhoods, although ekNN is parameter-free, its computational time is much more than classic kNN.

These comparisons of different neighborhoods provide us a practical guide to select the neighborhood type in SDPP. Considering the robustness of kNN neigbhorhood, in the remaining parts of this thesis, unless stated otherwise, we always use kNN base on its heuristic as the strategy for neighborhood selection.



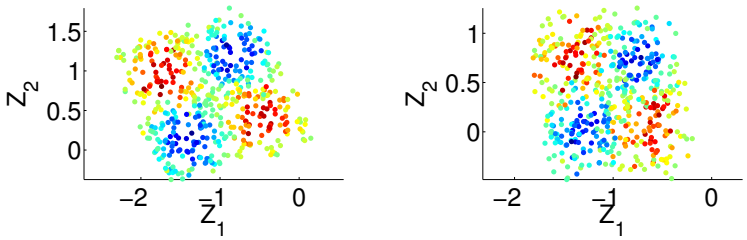
(a) $k = 3$ (b) $k = \text{round}(\log(n)) = 6$

(c) $k = 12$ (d) $k = 24$

(e) $\varepsilon = 0.21$ (f) $\varepsilon = (\log(n)/n)^{1/d} = 0.42$

(g) $\varepsilon = 0.84$ (h) $\varepsilon = 1.68$

Figure 3.5: Projection results based on not-enclosing neighborhoods.



(a) Projection based on ekNN (b) Projection based on NN

Figure 3.6: Projection results based on enclosing neighborhoods.

3.4 Kernel extension of SDPP

Generally, the performance of supervised dimensionality reduction by a linear projection may degrade in cases with a nonlinear mapping between the inputs and responses. In this section, we extend SDPP into kernelized version by the usage of kernel trick. The important intuition of the kernel trick is to map the data from the original input space to another higher (even infinite) dimensional feature space, $\phi : \mathcal{X} \mapsto \mathcal{F}$, and then perform a linear projection in this new feature space.

Denote the kernel matrix $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$. If we denote the matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1), \ldots, \boldsymbol{\phi}(\mathbf{x}_n)]$, we have $\mathbf{K} = \boldsymbol{\Phi}^T \boldsymbol{\Phi}$. Before proceeding, the mapped data in the feature space needs to be centered as follows. For the $n \times n$ kernel matrix of training data, $\mathbf{K}$, we have

$$\mathbf{K} \leftarrow (\mathbf{I} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T)\mathbf{K}(\mathbf{I} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T).$$

where $\mathbf{I}$ is an n-dimensional identity matrix, $\mathbf{1}_n$represent the vectors whose elements are ones with length n. respectively.

Based on the representation theory [1], assuming the projection matrix $\mathbf{W} = \boldsymbol{\Phi}\boldsymbol{\Omega}$, then we have the squared distance in the reduced feature space,

$$\begin{aligned} d_{ij}^2 &= (\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\phi}(\mathbf{x}_j))^T \mathbf{W}\mathbf{W}^T (\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\phi}(\mathbf{x}_j)) \\ &= (\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\phi}(\mathbf{x}_j))^T \boldsymbol{\Phi}\boldsymbol{\Omega}\boldsymbol{\Omega}^T\boldsymbol{\Phi}^T (\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\phi}(\mathbf{x}_j)) \\ &= (\boldsymbol{\Phi}^T\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\Phi}^T\boldsymbol{\phi}(\mathbf{x}_j))^T \boldsymbol{\Omega}\boldsymbol{\Omega}^T (\boldsymbol{\Phi}^T\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\Phi}^T\boldsymbol{\phi}(\mathbf{x}_j)) \\ &= (\mathbf{K}_i - \mathbf{K}_j)^T \boldsymbol{\mathcal{P}} (\mathbf{K}_i - \mathbf{K}_j), \end{aligned} \tag{3.17}$$

where PSD matrix $\boldsymbol{\mathcal{P}} = \boldsymbol{\Omega}\boldsymbol{\Omega}^T$ is with the size $n \times n$ and $\mathbf{K}_i$ is the ith column of the kernel matrix $\mathbf{K}$. Note that the square of the distance for the responses in the feature space is

$$\delta_{ij}^2 = \mathbf{K}_{ii}^y + \mathbf{K}_{jj}^y - 2\mathbf{K}_{ij}^y, \tag{3.18}$$

where $\mathbf{K}^y$ is the kernel matrix for response space.

The optimization of kernelized SDPP is same with original SDPP, replacing $\mathbf{X}$ and $\mathbf{W}$ by $\mathbf{K}$ and $\mathbf{\Omega}$, respectively. Moreover, the out-of-sample projections under the kernelized SDPP is $\mathbf{Z}_{\text{test}} = \mathbf{\Omega}^T\mathbf{K}_{\text{test}}$, where $\mathbf{K}_{\text{test}}$ is the kernel matrix between test points and training points and it should be centered before the projection,

$$\mathbf{K}_{\text{test}} \leftarrow (\mathbf{K}_{\text{test}} - \frac{1}{n}\mathbf{1}_{n_t}\mathbf{1}_n^T)(\mathbf{I} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T),$$

where n_t is the number of test points.

In some application areas such as image processing and genomics, the dimensionality of the data d is much larger than the number of data points (i.e., $d \gg n$). In these situations, applying SDPP is impractical due to the large PSD matrix with the size $d \times d$. Favourably, another important benefit of kernel SDPP is that it can reduce the computational complexity by learning a much smaller projection matrix $\mathbf{\Omega}$ with the size $n \times n$.

Part II

Experiments

Chapter 4

Experimental Evaluation

In this section, we firstly apply SDPP on both synthetic and real-world data sets to demonstrate its effectiveness as a tool of dimensionality reduction for regression. We compare our algorithm with other state-of-the-art dimensionality reduction methods, including PLS, SPCA and KDR. Secondly, we extend SDPP to make it suitable for classification tasks. Compared with FDA, SPCA and KDR, SDPP is applied on several real-world data sets for classification.

In this section, unless stated otherwise, we always use kNN as the strategy to construct the neighborhood graph, and number of k is selected as $k = \text{round}(\log(n))$, where n is the number of training points.

4.1 SDPP for Regression

Both synthetic and real-world data sets are considered for evaluating SDPP in regression tasks. The data sets used in this section are described briefly in Table 4.1.

Data sets	d	r	Size of training set	Size of test set
Curve line	10	1	500	1500
Servo	4	1-4	111	56
Tecator-fat	100	1-4	143	72
Auto-price	15	1-4	106	53
Ducks	4096	1	62	10

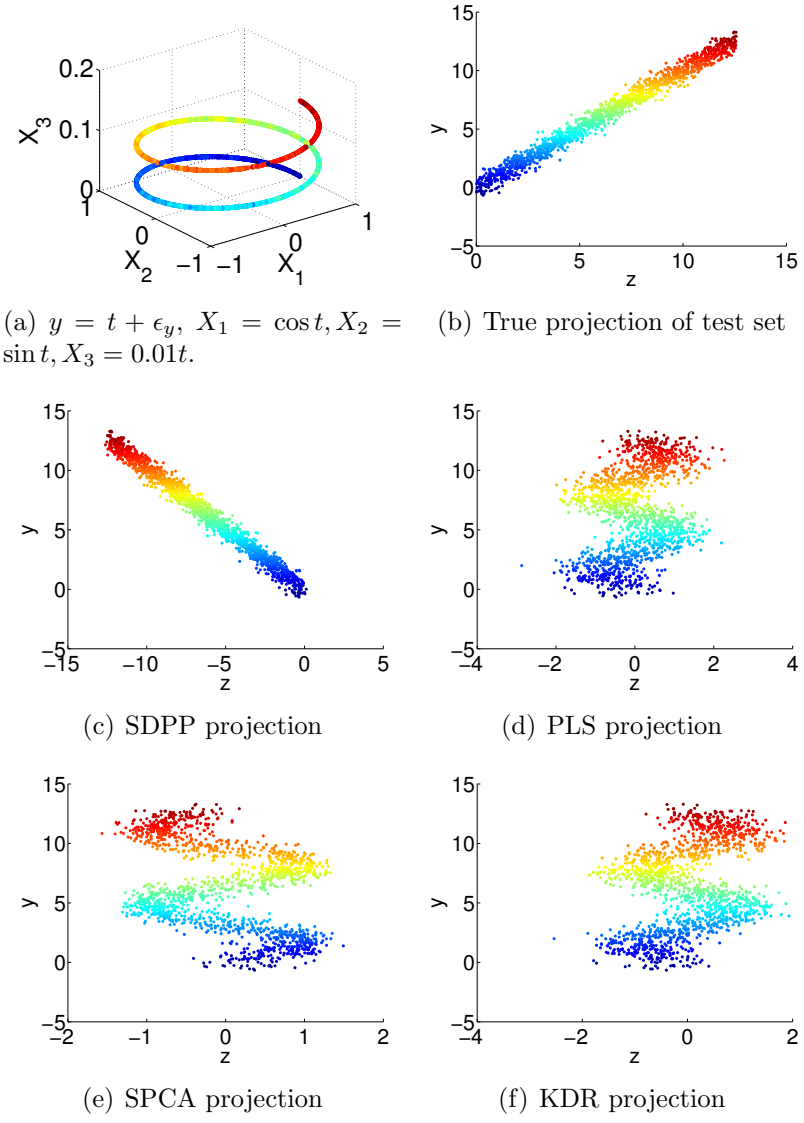
Table 4.1: Description of the data sets for regression tasks.

4.1.1 Curve line

The curve line data set is constructed as follows by a latent variable t uniformly distributed in $[0, 4\pi]$,

$$\mathbf{x} = [\cos t, \sin t, 0.01t, \boldsymbol{\epsilon}_{\mathbf{x}}]^T$$
$$y = t + \epsilon_y,$$

where $\boldsymbol{\epsilon}_{\mathbf{x}}$ is a seven-dimensional vector as input variables, $\boldsymbol{\epsilon}_{\mathbf{x}} \sim N(0, \mathbf{I}_7)$, thus $\mathbf{x} \in \mathbb{R}^{10}$, and ϵ_y is the noise term for the response, $\epsilon_y \sim N(0, 1)$. Figure 4.1(a) is the plot of three useful features of this data set associated with their responses indicated by different colors. We can observe that the underlying manifold is determined by the latent variable t along a straight line, as plotted in Figure 4.1(b). We generate 2000 data points, and use 500 for training and the remaining for test.



(a) $y = t + \epsilon_y$, $X_1 = \cos t, X_2 = \sin t, X_3 = 0.01t$.

(b) True projection of test set

(c) SDPP projection

(d) PLS projection

(e) SPCA projection

(f) KDR projection

Figure 4.1: Projection comparison on curve line data set.

Figure 4.1(c-f) presents the one-dimensional projections of the four methods on the test set, where x-axis is the projected variable z and its true response is along the y-axis. SDPP successfully unfolds this manifold structure along a straight line while the other three methods fail. We list the projection matrix $\mathbf{W} \in \mathbb{R}^{10\times 1}$ below,

obtained from the four methods.

$$\mathbf{W}_{SDPP} = [-0.04,\ -0.01,\ \underline{-99.14},\ 0.06,\ -0.04,\ 0.07,\ -0.05,\ 0.02,\ -0.08,\ -0.01]^T,$$
$$\mathbf{W}_{PLS} = [0.05,\ \underline{-0.83},\ \underline{0.13},\ 0,\ 0.04,\ \underline{0.25},\ \underline{0.17},\ \underline{0.27},\ \underline{-0.29},\ \underline{-0.22}]^T,$$
$$\mathbf{W}_{SPCA} = [-0.10,\ \underline{0.98},\ -0.04,\ -0.03,\ 0.03,\ -0.06,\ \underline{-0.13},\ -0.02,\ \underline{0.11},\ 0.01]^T,$$
$$\mathbf{W}_{KDR} = [0.10,\ \underline{-0.93},\ \underline{-0.14},\ 0,\ 0.07,\ \underline{0.16},\ 0.09,\ \underline{0.12},\ \underline{-0.17},\ \underline{-0.15}]^T.$$

We underline the weights with absolute value larger than 0.10 for comparison. SDPP weights the third feature most and ignores the other ones, thus, learns the correct projection direction. However, PLS, SPCA and KDR are not capable to identify the correct projection directions and even assign large weights to the noisy features X_4 to X_{10}.

4.1.2 Predicting rotation angles on an image manifold

In this experiment, we apply our approach to predict rotation angles on an image manifold. The real-world data set is a collection of duck images with different rotation angles in 3D scene, selected from Amsterdam Library of Object Images (ALOI) [1] image database. The ducks data set contains 72 images with equally spaced rotation angles, $[0°, 5°, \ldots, 355°]$. The original size of each image is 192×144 pixels. To reduce the computational complexity, we resize the original images and crop them into a suitable size of 64×64. Figure 4.2 shows four sample images of this data set. We select 10 equally spaced images for test and the remaining 62 images for training.

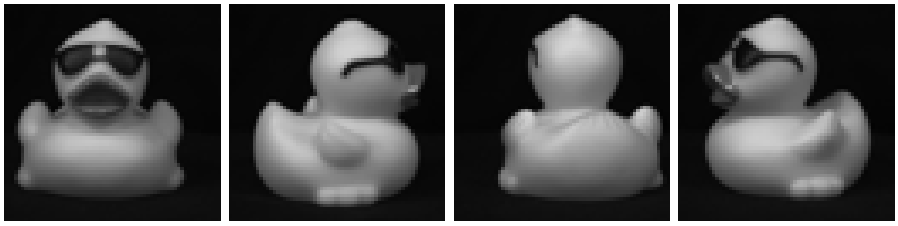


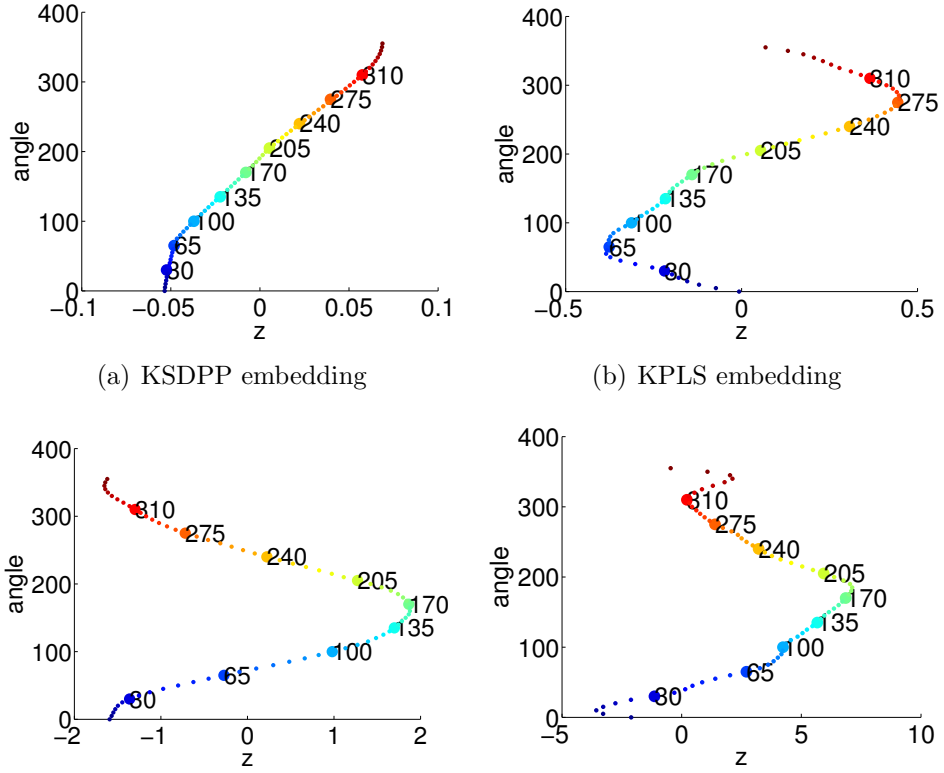
Figure 4.2: Four sample images of rotating ducks

Here we care about the rotation angles and aim to embed the high-dimensional images into a low-dimensional space that can unveil this aspect of image variations. Clearly, the dimensionality of each image is much larger than the number of samples, i.e. $d \gg n$. It is impractical to employ non-kernelized SDPP for two reasons. The first one is that we cannot learn a large projection matrix $\mathbf{W} \in \mathbb{R}^{4096 \times r}$ accurately, from only tens of samples. The second reason is the low learning speed for such a large projection matrix. To overcome this problem, the kernelized SDPP is applied for this task. Also kernel PLS and kernel SPCA are used for comparison. Since there

[1] http://staff.science.uva.nl/ aloi/

is no kernelized version for KDR, we only use KDR for this task.

In our experiment, RBF kernel is used in all the four methods, $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2})$. The kernel width is chosen by cross-validation around the median value of the pairwise Euclidean distances among all the data samples. Based on our experimental validation, this strategy of setting the kernel width works effectively for these four approaches.



(a) KSDPP embedding

(b) KPLS embedding

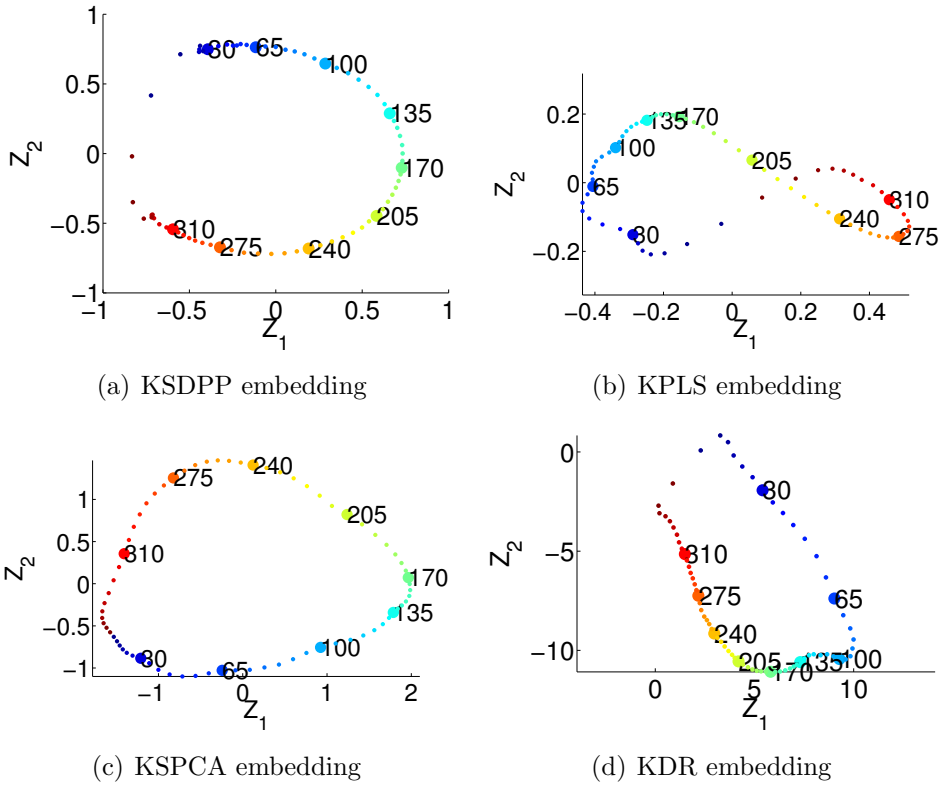
(c) KSPCA embedding

(d) KDR embedding

Figure 4.3: One-dimensional embedding of the ducks data set.

We plot the one-dimensional embeddings of the four methods in Figure 4.3, where x-axis represents the one-dimensional embedding, y-axis is the rotation angle. Small and large dots are used to indicate the training and test images, respectively. To be more clear, the test images are marked by their measured rotation angles. The plot in Figure 4.3(a) yielded by KSDPP reveals a clear linear relation between the projections and the rotation angles. Therefore, if we regress the angles on the low-dimensional representations, a linear regression function is appropriate and likely to be efficient. From this point of view, KSDPP successfully unfolds this image manifold for the regression task. In the embeddings from the other methods, a key observation is that the same embedding point might correspond different rotation angles which violates the definition of a *surjective* function. The violation can cause difficulties for an accurate regression in the low-dimensional space because in regression we assume the mapping from the input space to the output space are surjective. Hence, KPLS,

KSPCA and KDR fail to yield appropriate embeddings for regression.



(a) KSDPP embedding

(b) KPLS embedding

(c) KSPCA embedding

(d) KDR embedding

Figure 4.4: Two-dimensional embedding of the ducks data set.

We also plot the two-dimensional embeddings of this data set in Figure 4.4. KSDPP obtains smooth embeddings of the ducks with different rotation angles. The embeddings of KSDPP and KDR show a larger gap in the transition from 355° to 0° due to the significant different between their rotation angles. KPLS produces an unfaithful embeddings due to the presence of the tangle of the embeddings.

4.1.3 Regression on UCI data sets

Three commonly-used data sets from UCI repository [18], Servo, Tecator-fat and Auto-price, are used for experimental evaluation. The description of the data sets are shown in Table 4.1. All the data sets are preprocessed by mean centering and normalizing to the unit variance. We randomly split each data set into two parts, two thirds for training and the remaining for test. The experiments are run ten times according to the different splitting of the training and test sets. After the step of dimensionality reduction, a simple linear regression model is applied.

Table 4.2 depicts the RMSE ("mean $\pm$ std") on the test sets for each method with respect to different reduced dimensions. For each value of r, we highlight the minimal RMSE by boldface numbers. In most cases, SDPP exhibits an equivalent or better generalization performance compared with other three methods. Advantageously, the

performance of SDPP does not change too much with respect to the altering of projection dimensions. This behaviour is beneficial in practical applications when the projection dimension needs to be determined laboriously by cross-validation. Moreover, SDPP performs more stably with the random splitting of the data sets since the standard deviation of RMSE is less than that of other methods.

Data sets	r	SDPP	PLS	SPCA	KDR
Servo	1	1.3283±0.0748	1.3555±0.1194	1.3852±0.1194	**1.1946±0.1083**
	2	**1.2070±0.1140**	1.2432±0.1568	1.3264±0.1495	1.2308±0.1475
	3	1.1826±0.1344	1.1719±0.1305	1.2879±0.1581	**1.1592±0.1179**
	4	**1.1733±0.1315**	**1.1733±0.1315**	**1.1733±0.1315**	1.6181±0.1427
Tecator-fat	1	**2.6552±0.3943**	6.9072±0.5501	7.2070±0.5689	5.6043±2.1468
	2	**2.5606±0.3286**	5.4518±0.4608	5.7623±0.4530	3.3827±1.6056
	3	2.4589±0.1870	2.3142±0.1795	5.0767±0.9667	**2.1919±0.1645**
	4	**2.2061±0.1472**	2.2071±0.2393	**2.1791±0.2192**	2.2523±0.2812
Auto-price	1	2.8772±0.3536	2.8733±0.3405	2.9272±0.3173	**2.7282±0.3909**
	2	**2.6987±0.3750**	2.7205±0.3912	2.9043±0.2737	2.7680±0.4145
	3	**2.6757±0.3728**	2.6978±0.3941	2.7348±0.3399	2.8321±0.3362
	4	**2.6955±0.3753**	2.7033±0.3823	**2.6901±0.3937**	2.7853±0.3937

Table 4.2: RMSE ("mean ± std") on the test sets of three data, Servo, Tecator-fat and Auto-price.

4.2 SDPP for Classification

In SDPP formulation for regression in Eq. 3.1, the criterion sets the dissimilarity between two output responses (i.e., δ_{ij}) as their Euclidean distance. To extend the SDPP for classification tasks, here we redefine this dissimilarity by a binary set $\{0, 1\}$ to indicate their difference of class labels. That is, if two data points belong to the different classes, $\delta_{ij} = 1$, otherwise, $\delta_{ij} = 0$. This simple extension facilitates SDPP for dimensionality reduction in classification.

4.2.1 Tai Chi

Figure 4.5 presents the well-known Tai Chi model in the Asian culture. The black and white regions indicate two classes Yin and Yang, respectively. The concepts of Yin and Yang provide the intellectual framework for much of ancient Chinese scientific development especially in fields like biology and traditional Chinese medical science [16]. As illustrated by the Tai Chi figure, the exercise of Yin and Yang is the foundation of the entire universe. Yang, the element of light and life (in white), and Yin, the element of darkness and death (in black), are the most natural and original type of classes. Originated from each other, they represent all possible kinds of opposite forces and creatures, yet Yin and Yang work with each other in proper harmony. This makes the understanding and separation of them a difficult binary discriminant problem in real life applications.

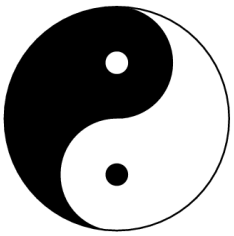


Figure 4.5: Tai Chi with two classes Yin and Yang.

The basic structure of Tai Chi is formed by drawing one large circle, two medium half circles and two small circles. The two small Yin and Yang circles located at the centers of the Yang and Yin half circles that are tangent to each other and also to the large circle. The Tai Chi model can be simulated as follows:

- X_1 and X_2 distribute uniformly from within the large circle. We then assign the class label $y = 1$ and $y = 2$ to the points located in the Yin area and the Yang area accordingly, see Figure 4.2.1.
- X_3, X_4, X_5 are the noisy features following the distribution $N(0, \mathbf{I}_3)$.

Thus, the original input $\mathbf{x}$ is a five-dimensional vector. The goal is to identify the first two effective directions for classification. We generate 2000 points and use the first 500 of them for training and the remaining for test. Four methods, SDPP, Fisher

discriminant analysis, SPCA and KDR are employed for this task. Figure 4.7 presents the two-dimensional projections for the test set. Clearly, SDPP finds the two correct projection directions successfully while the other three methods fail.

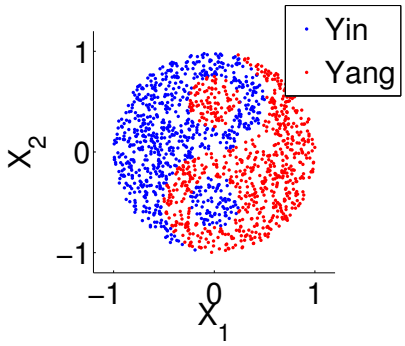
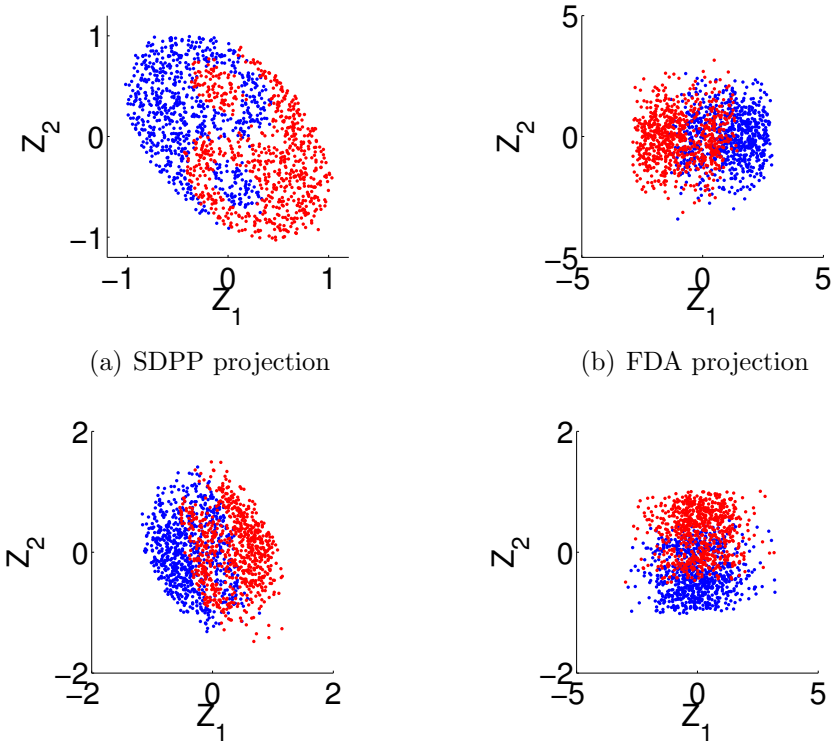


Figure 4.6: Simulation of Tai Chi model.



(a) SDPP projection

(b) FDA projection

(c) SPCA projection

(d) KDR projection

Figure 4.7: Two-dimensional projection of the Tai Chi data for the test set.

4.2.2 Classification on UCI data sets

Now we apply SDPP in some of UCI data sets to demonstrate its effectiveness in classification tasks. Compared with SDPP, three other methods, Fisher discriminant analysis, SPCA and KDR, are also used for supervised dimensionality reduction. The

description of these UCI data sets is presented in Table 4.3.

Data sets	d	r	Number of class	Size of training set	Size of test set
Glass	10	1-4	6	143	71
Segmentation	18	1-4	7	140	70
Isolet	617	1-4	26	1039	520

Table 4.3: Description of data sets for classification tasks.

For each data set, we use two thirds of the data for training and the remaining for test. Ten independent runs are implemented based on different splitting of the training and test sets. For the data set Isolet with a high dimensionality, we apply PCA to reduce the dimensionality into 30 as a preprocessing step. Since the size of this data set is large, we do not implement KDR on it due to the computational issue. After the projections according to different values of dimension r, k-nearest neighbor classifier is applied for classification in the new subspaces. We set the number of nearest neighbors in kNN classifier as $k_c = C+1$, where C is the number of classes for each data set.

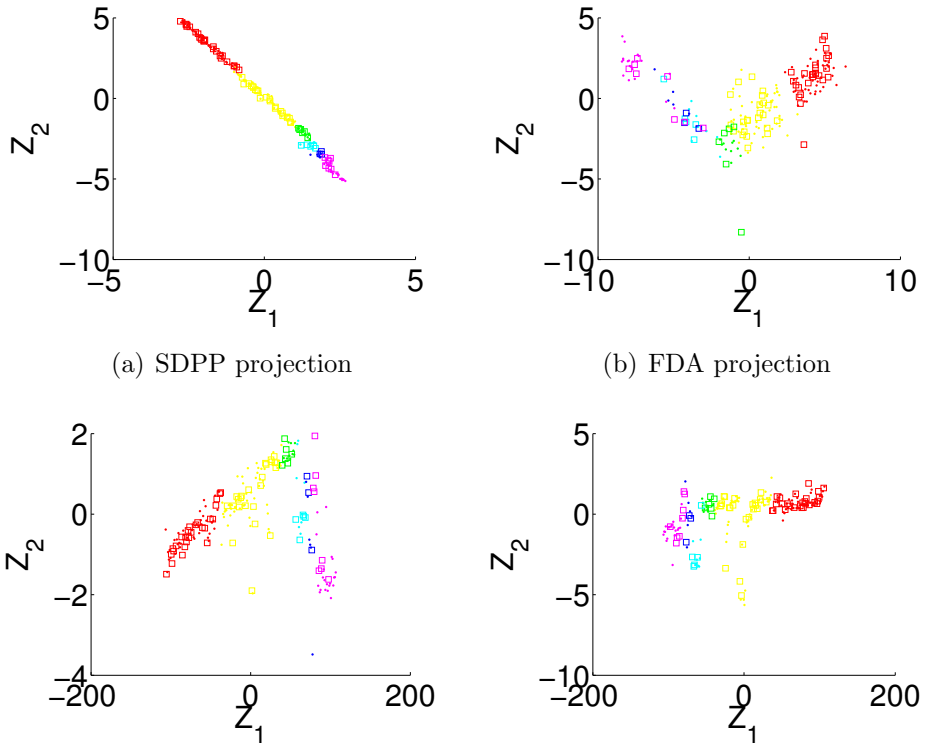
Table 4.4 shows the classification accuracies of the four methods on the test sets, in the form of "mean $\pm$ std". On the Glass data set, all the four methods exhibit the effectiveness for this classification task. SDPP provides slightly better results than the other three methods. For the Segmentation and Isolet data, SDPP and FDA performs better than the others.

For the purpose of visualizing these three data sets, we plot the two-dimensional projections for both the training and test samples. For the Glass data, Figure 4.8 shows the projections of the four methods, where dots ($\cdot$) and squares ($\square$) are used to distinguish the training and test samples. SDPP yields a better projection because all the classes are clearly separated and there are no points of mixing between different classes. The Segmentation data set is more difficult for the four methods to separate all the seven classes clearly, as seen in Figure 4.9. There are four classes mixing with each

Data sets	r	SDPP	FDA	SPCA	KDR
Glass	1	**99.58±0.68**	99.15±0.98	99.30±1.04	99.44±0.98
	2	**99.72±0.59**	98.87±0.89	99.44±0.98	99.44±0.98
	3	**99.58±0.68**	98.73±1.04	99.44±0.98	99.44±0.98
	4	**99.58±0.68**	98.59±1.15	99.44±0.98	99.44±0.98
Segmentation	1	**91.14±5.38**	86.29±3.76	74.29±4.95	83.43±6.03
	2	81.14±4.09	**89.86±3.12**	76.29±5.80	85.14±6.91
	3	91.14±5.98	**93.57±2.36**	81.57±5.53	83.57±5.05
	4	**98.29±1.13**	97.00±2.07	82.71±4.88	84.57±5.63
Isolet	1	93.70±1.23	**93.77±0.91**	93.34±1.24	-
	2	**97.69±1.06**	97.09±0.89	94.13±1.02	-
	3	98.46±0.62	**98.73±0.44**	96.96±0.59	-
	4	**98.89±0.46**	98.68±0.67	97.62±0.66	-

Table 4.4: Classification accuracies by percentage on the test sets of data, Glass, Segmentation and Isolet.

other in all the four projections. However, KDR obtains the best separation. Lastly, on the data Isolet, Figure 4.10-4.12 present the projection results of SDPP, FDA and SPCA, respectively. Both SDPP and FDA produce a more separated projection while SPCA tends to congregate different classes together.



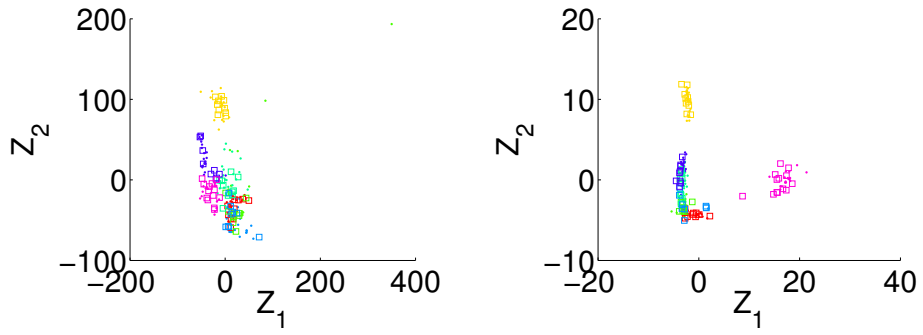
(a) SDPP projection

(b) FDA projection

(c) SPCA projection

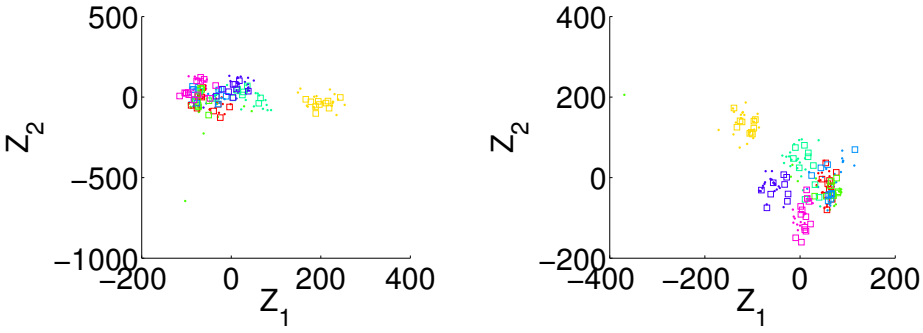
(d) KDR projection

Figure 4.8: Two-dimensional projection of the Glass data for the test set.



(a) SDPP projection

(b) FDA projection



(c) SPCA projection

(d) KDR projection

Figure 4.9: Two-dimensional projection of the Segmentation data for the test set.

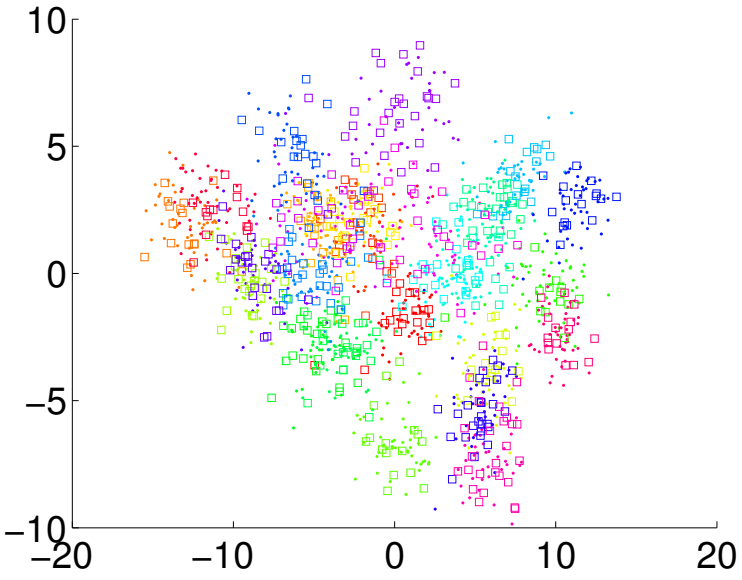


Figure 4.10: Two-dimensional projection of SDPP on the Isolet data for the test set.

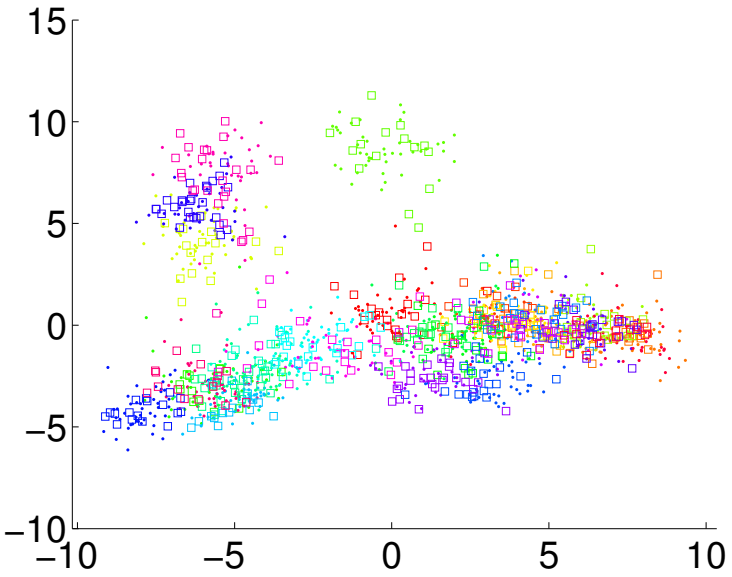


Figure 4.11: Two-dimensional projection of FDA on the Isolet data for the test set.

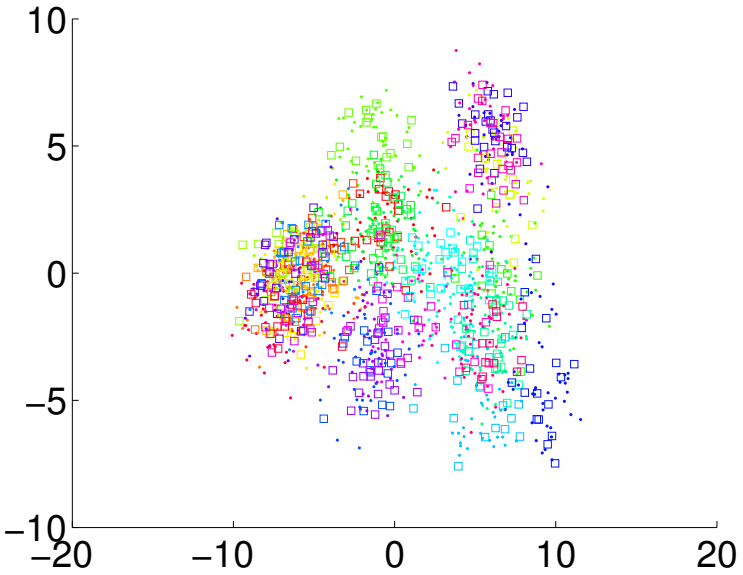


Figure 4.12: Two-dimensional projection of SPCA on the Isolet data for the test set.

Chapter 5

Conclusions

This work presents a novel supervised dimensionality reduction method, especially for regression tasks named Supervised Distance Preserving Projection (SDPP). Motivated by continuity preservation for a continuous regression function, SDPP casts the goal of continuity preservation into the match of local geometry between the low-dimensional subspace and the response space. SDPP finds an affine transformation such that the difference between local pairwise distances in the subspace and those in the response space are minimized. We have provided a geometrical interpretation on the criterion of SDPP and several advantages of SDPP were also presented, such as the capability of handling multiple outputs and out-of-sample data points.

Three optimization schemes, SDP, SQLP and conjugate-gradient optimization, were introduced for solving the SDPP efficiently. Conjugate-gradient optimization for SDPP scales well with the dimensionality of the original data while SDP and SQLP fail. To evaluate the performance of SDPP in different parameter settings, we employed two categories of strategies for neighborhood selection, not-enclosing neighborhood and enclosing neighborhood. We also derived a kernelized version of SDPP to deal with nonlinear data. Kernel SDPP can reduce the computational complexity of DR tasks where the dimensionality of the input covariates is much larger than the number of data samples. An intuitive extension of SDPP is also presented suitable for classification tasks.

The experimental evaluation has demonstrated the effectiveness of SDPP and its kernelized version on dimensionality reduction for regression and classification tasks. Various synthetic and real-world data sets are used for the evaluation. Empirically, SDPP and its kernelized form have shown their superiority on the task of supervised dimensionality reduction compared with other existing state-of-the-art techniques.

To summarize, our approach of supervised dimensionality reduction has several appealing merits. Firstly, for dimensionality reduction in regression, to our knowledge, we are the first to propose to preserve continuity of a regression function. This preservation guarantees that the regression information between the original inputs and outputs are retained as much as possible. Secondly, our criterion is based on the local geometry of the input space and local pairwise distances are used to describe

the local geometry. Thus, our method is capable of reducing the dimensionality only provided pairwise distance of the training set, without the pairs of the inputs and outputs. Finally, the optimization schemes are efficient for handling extremely high-dimensional data.

With regard to the future work and extension of SDPP, there are several possibilities to investigate.

- In the criterion of SDPP, for each data point, we only consider the points inside its neighborhood and try to make them stay close after projection. From the view of energy models [39], this can be seen as a *attraction* force in the neighborhood. SDPP ignores the points that do not belong to the neighborhood. Therefore, adding the *repulsion* force to the criterion for the points which are not inside the neighborhood can be a promising improvement for SDPP.

- SDPP considers the local match between geometrical structure of the low-dimensional subspace and output space. In this work, we use the squared Euclidean distance as a divergence to quantify this match. Other types of divergences, such as Kullback-Leibler divergence, Itakura-Saito divergence [12], or a more general one Bregman divergence [37], are potential choices to apply to capture more statistically meaningful projections.

- To extend SDPP for controlling the solution of the projection matrix, some regularization terms are possible to add to the criterion of SDPP, such as l_1-norm to yield a sparse solution for feature selection [52] or $l_{2,1}$-norm to weight the features across all data points with joint sparsity [38].

- It is also possible to extend SDPP under the semi-supervised setting, where both the data samples with responses (or labels) and the ones without responses (or labels) are given. The manifold regularization [5] can be an alternative in this case.

Bibliography

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

[2] E. Barshan, A. Ghodsi, Z. Azimifar, and M. Zolghadri Jahromi. Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds. *Pattern Recognition*, 44:1357–1371, 2010.

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

[4] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems*, 1:585–592, 2002.

[5] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[6] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1966.

[7] A. Ben-Tal and A.S. Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*, volume 2. Society for Industrial Mathematics, 2001.

[8] C.M. Bishop. *Pattern recognition and machine learning*. Springer New York, 2006.

[9] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[10] M.R. Brito, E.L. Chavez, A.J. Quiroz, and J.E. Yukich. Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. *Statistics & Probability Letters*, 35(1):33–42, 1997.

[11] F.R.K. Chung. *Spectral graph theory*. American Mathematical Society, 1997.

[12] J.F. Coeurjolly, R. Drouilhet, and J.F. Robineau. Normalized information-based divergences. *Problems of Information Transmission*, 43(3):167–189, 2007.

[13] R.D. Cook and X. Yin. Dimension reduction and visualization in discriminant analysis. *Australian and New Zealand Journal of Statistics*, 43(2):147–199, 2001.

[14] P. Demartines and J. Hérault. Cca: Curvilinear component analysis. In *Colloque sur le traitement du signal et des images, FRA, 1995*. GRETSI, Groupe dEtudes du Traitement du Signal et des Images, 1995.

[15] D.L. Donoho. High-dimensional data analysis: The curses and blessings of dimensionality. *AMS Math Challenges Lecture*, pages 1–32, 2000.

[16] P.B. Ebrey. *Chinese civilization: a sourcebook*. Free Pr, 1993.

[17] J. Fan and Y. Fan. High dimensional classification using features annealed independence rules. *Annals of statistics*, 36(6):2605, 2008.

[18] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[19] K. Fukumizu, F.R. Bach, and M.I. Jordan. Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *Journal of Machine Learning Research*, 5:73–99, 2004.

[20] K. Fukumizu, F.R. Bach, and M.I. Jordan. Kernel dimension reduction in regression. *Annals of Statistics*, 37:1871–1905, 2009.

[21] A. Globerson and S. Roweis. Metric learning by collapsing classes. *Advances in Neural Information Processing Systems*, 18:451, 2006.

[22] T. Graepel. Kernel matrix completion by semidefinite programming. pages 141–142, 2002.

[23] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf. Measuring statistical dependence with Hilbert-Schmidt norms. In *Algorithmic learning theory*, pages 63–77. Springer, 2005.

[24] A. Gretton, K. Fukumizu, C.H. Teo, L. Song, B. Schölkopf, and A. Smola. A kernel statistical test of independence. In *Advances in Neural Information Processing Systems*, 2008.

[25] M. R. Gupta, E. K. Garcia, and E. Chin. Adaptive local linear regression with application to printer color management. *IEEE Transactions on Image Processing*, 17:936–945, 2008.

[26] A. Höskuldsson. PLS regression methods. *Journal of Chemometrics*, 2(3):211–228, 1988.

[27] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[28] Q. Zhu K.Q. Weinberger, F. Sha and L.K. Saul. Graph laplacian regularization for large-scale semidefinite programming. In *Advances in neural information processing systems*, pages 1489–1496, 2006.

[29] J.A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer Verlag, 2007.

[30] B. Li, H. Zha, and F. Chiaromonte. Contour regression: A general approach to dimension reduction. *The Annals of Statistics*, 33(4):1580–1616, 2005.

[31] K.C. Li. Sliced inverse regression for dimension reduction. *Journal of the American Statistical Association*, 86(414):316–327, 1991.

[32] K.C. Li. On principal hessian directions for data visualization and dimension reduction: Another application of Stein's lemma. *Journal of the American Statistical Association*, 87(420):1025–1039, 1992.

[33] Z. Li and J. Liu. Constrained clustering by spectral kernel learning. In *Internal Conference of Computer Vision*, 2009.

[34] E.O. Postma L.J.P. van der Maaten and H.J. van den Herik. Dimensionality reduction: A comparative review. Technical report, Tilburg University Technical Report.

[35] J. Löfberg. Yalmip : A toolbox for modeling and optimization in matlab. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[36] S. Mika, G. Ratsch, J. Weston, B. Schölkopf, and KR Mullers. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pages 41–48. IEEE, 1999.

[37] N. Murata, T. Takenouchi, T. Kanamori, and S. Eguchi. Information geometry of U-Boost and Bregman divergence. *Neural Computation*, 16(7):1437–1481, 2004.

[38] F. Nie, H. Huang, X. Cai, and C. Ding. Efficient and robust feature selection via joint $l_{2,1}$-norms minimization. In *Advances in Neural Processing Systems*, 2010.

[39] A. Noack. Energy models for graph clustering. *Journal of Graph Algorithms and Applications*, 11(2):453–480, 2007.

[40] M.D. Penrose. A strong law for the longest edge of the minimal spanning tree. *Annals of Probability*, 27:246–260, 1999.

[41] R. Rosipal and N. Krämer. Overview and recent advances in partial least squares. *Subspace, Latent Structure and Feature Selection*, pages 34–51, 2006.

[42] R. Rosipal and L.J. Trejo. Kernel partial least squares regression in reproducing kernel hilbert space. *Journal of Machine Learning Research*, 2:97–123, 2002.

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

[44] J.W. Sammon Jr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 100(5):401–409, 1969.

[45] B. Schölkopf, A. Smola, and K.R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[46] D.W. Scott and J.R. Thompson. Probability density estimation in higher dimensions. In *Computer Science and Statistics: Proceedings of the Fifteenth Symposium on the Interface*, volume 528, pages 173–179. North-Holland, Amsterdam, 1983.

[47] Fei Sha and Lawrence K. Saul. Analysis and extension of spectral methods for nonlinear dimensionality reduction. In *In Proceedings of the Twenty Second International Conference on Machine Learning (International Conference on Machine Learning*, pages 785–792, 2005.

[48] R. Sibson. A brief description of natural neighbors interpolation. In V. Barnett, editor, *Interpreting Multivariate Data*, pages 21–36. Wiley, New York, 1981.

[49] L. Song, A. Smola, K. Borgwardt, and A. Gretton. Colored maximum variance unfolding. In *Advances in neural information processing systems*, 2008.

[50] J.F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1):625–653, 1999.

[51] J.B. Tenenbaum, V. Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319, 2000.

[52] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[53] K.C. Toh, M.J. Todd, and R.H. Tutuncu. Sdpt3-a matlab software package for semidefinite programming. *Optimization Methods and Software*, 11(12):545–581, 1999.

[54] R.H. Tütüncü, K.C. Toh, and M.J. Todd. Solving semidefinite-quadratic-linear programs using sdpt3. *Mathematical Programming*, 95(2):189–217, 2003.

[55] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.

[56] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[57] J.A. Wegelin. A survey of partial least squares PLS methods, with emphasis on the two-block case. Technical report, Seattle: Department of Statistics, University of Washington, 2000.

[58] K.Q. Weinberger, J. Blitzer, and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems*, 2006.

[59] K.Q. Weinberger, F. Sha, and L.K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the 21st International Conference on Machine learning*, page 106. ACM, 2004.

[60] H. Wold. Soft modeling by latent variables: the nonlinear iterative partial least squares approach. *Perspectives in probability and statistics, papers in honour of MS Bartlett*, 9:520–540, 1975.

[61] X.M. Wu, A.M.C. So, Z. Li, and S.Y.R. Li. Fast graph laplacian regularized kernel learning via semidefinite–quadratic–linear programming. In *Advances in Neural Information Processing Systems*, 2009.

[62] E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. pages 521–528, 2003.

[63] Y.R. Yeh, S.Y. Huang, and Y.J. Lee. Nonlinear dimension reduction with kernel sliced inverse regression. *IEEE Transactions on Knowledge and Data Engineering*, 21:1590–1603, 2009.

[64] F. Zhang. *The Schur complement and its applications*. Springer Verlag, 2005.