Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Srikrishna Raamadhurai

Supervised Probability Preserving Projection (SPPP)

Master's Thesis
Espoo, October 16, 2014

Supervisor: Professor Juha Karhunen, Aalto University
Instructor: Francesco Corona, PhD. (Docent), Aalto University

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author: Srikrishna Raamadhurai

Title:
Supervised Probability Preserving Projection (SPPP)

Date:	October 16, 2014	**Pages:**	66
Major:	Machine Learning and Data Mining	**Code:**	T-61
Supervisor:	Professor Juha Karhunen, Aalto University		
Instructor:	Francesco Corona, PhD. (Docent), Aalto University		

Dimensionality Reduction (DR) is the process of finding a reduced representation of a data set according to some defined criteria. DR may be performed in both unsupervised and supervised settings. Several techniques have been proposed in the literature for unsupervised DR, where the aim is usually to preserve some intrinsic characteristics of the data without using the output information. In most cases they are preferred as a preprocessing step while some may end up with clustering or visualizations. While much focus has been on unsupervised methods, supervised techniques are preferred when every sample has its output information. Even though obtaining this information may be expensive for some tasks, this supports supervised methods in trying to avoid the curse of dimensionality where the space may be sparse. The output information allows the methods to focus on each points' real neighbors unlike unsupervised methods.

In this thesis we aim to develop a supervised DR technique called Supervised Probability Preserving Projection (SPPP) that operates on probabilistic relations between points. More specifically we learn a linear transformation matrix that maps the input samples on to a projection space where the differences between the probabilistic similarities of the input covariates and their responses are minimized, given a neighborhood function.

This thesis begins by suggesting three probabilistic neighborhood functions for a recently proposed method called Supervised Distance Preserving Projections (SDPP). Motivations from the experimental results on synthetic examples leads to the development and introduction of a novel technique called Supervised Probability Preserving Projection (SPPP). The formulation of SPPP and optimizations for three versions namely Gaussian, Heavy-tail and Linear are presented. The experiments indicate competitive performance of SPPP compared to recent state-of-the-art methods suggesting its use for both regression and classification tasks alike.

Keywords:	DR, SPPP, SDPP, Probabilities, Pairwise Distances
Language:	English

Acknowledgements

I thank my decorated supervisor Prof. Juha Karhunen for allowing me to do my thesis under his guidance. I am grateful to my advisor Dr. Francesco Corona for his untiring efforts throughout this thesis. He was always available and motivating including his meticulous approach to reviewing the drafts. I thank the Department of ICS for providing me a place in the lab and the IT administration team for their continuous support.

Over the period of my stay in Finland, I had the luxury of making some friends who will always remain treasured in my heart. The long technical and trivial discussions including sports strengthened our friendship. Their varied academic background always provided some new ideas and outlook to my problems. I cannot complete thanking them verbally, but would like to at least mention their names (in random order) Luiza Sayfullina, Ehsan Amid, Abdulmelik Nesru, Ermiyas Endale, Gökcen Eraslan, Basak Eraslan.

It is often difficult for a new student to get noticed in the department. Thanks to my fortune that I found Prof. Timo Honkela and his exciting research activities. This led to my acquaintance with Oskar Kohonen, PhD student. I received unconditional guidance from Oskar as we worked on a summer project leading to my first technical paper at COLING 2014, co-authoring with Teemu Ruokolainen, PhD student. As I write my thesis, I am currently working under Prof. Honkela as an Information Systems Specialist.

Having a good work-life balance is everyone's dream. Especially for a strict vegetarian like me to survive the cold in a foreign land is not easy. Thanks to my family and Skype for making all this easier for me. I thank my dad Mr.Raamadhurai for having garnered immense confidence in me to let me pursue my Master's studies abroad. A younger sister is the rarest of a gem one could have. At times of my low, I would take inspiration from her (Srividya) to move forward. I am grateful to all members in my family up to my grannies who have stood by me so long and finally the Almighty.

Espoo, October 16, 2014
Srikrishna Raamadhurai

Contents

1 Introduction **6**
- 1.1 Need for Dimensionality Reduction . . . 6
- 1.2 Dimensionality Reduction . . . 7
  - 1.2.1 Variable Subset Selection . . . 7
  - 1.2.2 Feature Extraction . . . 8
  - 1.2.3 Evaluating Dimensionality Reduction . . . 11
- 1.3 Highlights and Structure of the Thesis . . . 12

2 A brief look at SDPP **14**
- 2.1 Formulation . . . 14
- 2.2 Optimization . . . 16
- 2.3 Neighborhood Selection in SDPP . . . 17
- 2.4 Synthetic examples . . . 19
  - 2.4.1 Data sets . . . 19
  - 2.4.2 Regression: Parity data . . . 22
  - 2.4.3 Classification: Taijitu symbol . . . 22

3 Probabilistic Neighborhoods for SDPP **24**
- 3.1 Entropy and Cross entropy for setting variance in neighborhoods . . . 26
- 3.2 Student's t-distribution for neighborhoods . . . 28
- 3.3 Synthetic examples . . . 28
  - 3.3.1 Regression: Parity data . . . 28
  - 3.3.2 Classification: Taijitu Symbol . . . 29

4 Supervised Probability Preserving Projection (SPPP) **32**
- 4.1 Motivation . . . 32
- 4.2 Cost function . . . 34
- 4.3 Gradients . . . 35
  - 4.3.1 Gaussian . . . 35
  - 4.3.2 Heavy-tail . . . 38
  - 4.3.3 Linear . . . 39

4.4 Synthetic examples 41
4.4.1 Regression: Parity data 41
4.4.2 Classification: Taijitu Symbol 43

5 Experimental Verifications **44**
5.1 Experimental setup 44
5.2 Regression on UCI data 45
5.2.1 Discussions 51
5.3 Classification on UCI data 52
5.3.1 Discussions 57

6 Conclusions **58**

7 Appendix **60**
7.1 A brief overview of PLS, KPLS and KDR 60

Chapter 1

Introduction

The recent advent of affordable electronics and social networking platforms have been contributing to massive data accumulation. Most parts of this information such as text, video and audio are mainly in human understandable form. While this is good for empirical analyses, it makes for example discovering the current trend in Music, the top performers in some industry or the causal variables in a drug test analysis, a non-trivial task. This is because such data have dimensionality that run in cases into millions, needless to mention their storage and handling costs.

1.1 Need for Dimensionality Reduction

Consider for instance a scanned collection of historic newspaper articles accumulated over centuries. Researchers interested in studying the history of a country would have to browse through the articles manually. Instead if such data were available in machine readable form, browsing through the articles using advanced search techniques would make the study easier and informative.

Natural Language Processing (NLP) is a field of computer science that deals with methods for processing text data. Usually applying NLP techniques require representing text as vectors which involves creating a term document matrix. Every document is converted to a vector of unique words. Depending on the number of documents considered, the number of unique words (or dimensions) grow, creating an increasingly sparse matrix. In most cases the dimensionality runs into millions making analysis harder. Consequently effective means of reducing the dimensionality of the data is necessary.

As pointed out in [33] Dimensionality Reduction (DR) is usually an ill-posed problem since neither the true local geometry (manifold) nor the real intrinsic dimensionality of the data is known beforehand. Moreover the number of points required to explain every additional dimension grows exponentially making the space sparse. This is often referred to as '*the curse of dimensionality*'. Despite these limitations, researchers have published several promising techniques over the past few decades.

1.2 Dimensionality Reduction

Dimensionality Reduction can be solved using two different approaches, *Variable Subset Selection* and *Feature Extraction*.

1.2.1 Variable Subset Selection

Variable Subset Selection is defined as the process of identifying a subset of more informative variables in the data set that can perfectly fit a model. Variable Selection typically works under the assumption that the data set may contain many *redundant* variables. Hence the necessity to prune such variables in a systematic manner, allows us to fit a model whilst retaining a reduced set of variables. Variable Selection may be broadly divided into *Wrappers*, *Filters* and *Embedded* methods [7].

Wrappers

Wrapper techniques are simple yet effective methods of scoring and ranking the most influential variables in the data set that improve the predictor performance. To this end, efficient search strategies including forward, backward or nested subset selection have been proposed. Forward selection implies adding variables in a manner that improves the predictor performance. Backward selection instead tries to remove variables one by one. Even though these methods may appear like 'brute force' and time consuming, they provide a good starting point for small datasets or when some intuition about the data is already available. Both the approaches have complementary benefits and drawbacks. While a particular variable has already been chosen by Forward selection, evaluating the performance of the predictor without this variable is often time consuming since it involves re-running the tests over an exhaustive set of combinations. The typical arbitrary ordering of variables

poses a similar problem to Backward selection and nested subset as well. In cases where the dimensionality is large, Wrapper methods are known to be NP-hard [1].

Filters

Another method of Variable Selection called **Filters** offer to retain a subset of the variables normally preferred as a preprocessing step irrespective of the chosen predictor. The Filters are often used as a limitation imposed over the Wrapper-scored variables thereby identifying a subset to be used thereon. Such filters may use mutual information (MI) [31] or some other performance evaluation criterion such as area under ROC curves [11] and so on.

Embedded methods

Embedded methods are powerful Variable Selection techniques that are more specific to the predictor chosen. These class of methods have an objective function that is optimized, eventually finding a reduced representation to let us build a model. For a detailed overview of Variable selection methods, the reader is referred to [10].

1.2.2 Feature Extraction

Feature extraction is the process of reducing the dimensionality of the data in order to obtain a meaningful combination of variables. Often they are preferred as a preprocessing step that preserves the intrinsic characteristics of the data. A carefully extracted set of features can be expected to contain a similar amount of information as in the original while allowing us to use the reduced representation thereon. These are then called feature vectors which can be made use of for further analysis or modeling.

Types of Dimensionality Reduction

Dimensionality Reduction may be performed in two main settings, *Unsupervised* and *Supervised*. Unsupervised methods are those that do not use output information in their design. One would then have to assume that the data has the necessary information within itself to reveal its structure. On the other hand

when every sample is labeled, Supervised methods utilize the additional information to find a reduced representation.

Unsupervised Dimensionality Reduction

Often unsupervised methods are used as a preprocessing step ending up with a clustering or visualization. Based on the chosen metric-to-be-preserved, several methods have been proposed in the literature that perform well, given their limitations.

Principal Component Analysis (PCA) [19] is a well known unsupervised technique that finds a linear subspace mapping of the input data in which much of the variance is explained. PCA computes an eigenvalue decomposition on a full input covariance matrix thereby producing a linearly uncorrelated set of variables called principal components. The obtained principal components represent the decreasing order of variance in the data set subject to orthogonality. **Kernel PCA** (KPCA) [25] is a kernel extension of PCA that computes the principal eigenvectors of the kernel matrix instead of the covariance matrix. The inner product of data points computed using the kernel function, allows construction of nonlinear mappings in that kernel space. Another unsupervised method called **Multidimensional Scaling** (MDS) [30] minimizes the differences in the Euclidean distances computed in both the input and the reduced spaces.

Unlike MDS, techniques such as **ISOMAP** [29] and **Maximum Variance Unfolding** (MVU) [37] try to preserve pairwise distances measured along the manifolds called geodesic distances. ISOMAP uses a neighborhood graph constructed by joining k-nearest neighbors and applies a classical scaling. Unfortunately topological instabilities occurring due to erroneous short-circuiting in the neighborhood matrix could hamper the performance of ISOMAP. One solution to this problem is to remove nearest neighbors that violate local linearity of the neighborhood graph [24]. ISOMAP may also suffer from 'holes' in the manifold. This has been overcome by tearing, which means building a maximum subgraph with no loops anymore [16]. MVU on the other hand, learns a kernel neighborhood matrix by maximizing the pairwise geodesic distances allowed by the neighborhood. This method is said to 'unfold' the data manifold while retaining its pairwise distances.

Another flavor of neighborhood retaining methods are the more recently proposed techniques such as **Stochastic Neighbor Embedding** (SNE) [12], **t-distributed Stochastic Neighbor Embedding** (t-SNE) [32] and **Neighbor Retrieval Visualizer** (NeRV) [36]. These methods consider the probability of

being neighbors instead of pairwise distances. All the three methods aim to find a suitable embedding that retains close by neighbors with some probability.

Some other unsupervised methods such as **Self Organizing Map** (SOM) [15], **Local Linear Embedding** [23] and **Laplacian Eigenmap** [4] preserve topological relations within the data set. SOM is an Artificial Neural Networks (ANN) inspired technique based on competitive learning of neurons. A two dimensional map of predefined topology of nodes is setup and their corresponding weights are learned. A U-matrix (unified distance matrix) value of a node represents its relative distance from its closest neighbors. **Autoencoder** is also an ANN inspired technique that offer ways to compress or encode high dimensional data. In the simplest case an autoencoder is a feedforward, non-recurrent neural network that reconstructs its input samples and learns useful features from them which can be decoded to once again recover the input samples. Autoencoders are usually trained using backpropagation variants such as conjugate gradient or steepest descent, however they may at times be slow to converge. Consequently, initialising the weights of the network that approximate the final solution have been suggested, often called as pretraining [13]. For a detailed review of other unsupervised methods, the reader is referred to [33].

Supervised Dimensionality Reduction

While much work has been focused on unsupervised methods, fewer prominent techniques exist for supervised methods.

Partial Least Squares (PLS) [38] computes a linear relationship between a set of input covariates and their responses by maximizing the covariances on either space thereby obtaining an orthogonal transformation matrix. The Kernel version of PLS called **KPLS** was designed to handle nonlinear cases which maximizes the covariances in the kernel space [22].

Kernel Dimensionality Reduction (KDR) [8] is a linear DR method which computes an orthogonal transformation matrix that maximizes the conditional independence of the input samples and their responses given the subspace. This is performed in the Reproducing Kernel Hilbert Space (RKHS) wherein the conditional independence is imposed by minimizing the conditional covariance. A supervised version of PCA called **SPCA** [3] has been introduced which computes the principal components with maximum dependency on the responses. The optimization once again follows solving an Eigen problem, this time with a weighted

covariance matrix. The measure of dependence in SCPA is based on the Hilbert-Schmidt Independence Criterion (HSIC). The HSIC is zero if and only if two random variables are independent. A more recently introduced method called **Supervised Distance Preserving Projections** (SDPP) [39] learns a linear transformation matrix leading to a projection space where the differences in pairwise distances between the input covariates and their responses are minimized locally. SDPP operates on pairwise (Euclidean) distances and is motivated on the Weierstrass definition of continuity. This method uses a binary-valued neighborhood matrix to indicate the potential neighbors of points in both spaces.

1.2.3 Evaluating Dimensionality Reduction

It is in general impossible to retain all the information from the original space on to the reduced space. Eventally we are bound to lose some information which we try to minimize as much as possible. Two commonly used measures for evaluating the goodness of a projection are precision and recall. **Precision** is defined as the ability of a projection to project points closely in the reduced space provided they were neighbors in the original space. **Recall** is the ability of a projection to retain neighbors from the original space and project them closely in the reduced space.

It is often difficult to obtain both good precision and recall. For instance, imagine a 2D scattered set of points in some $X1, X2$ plane. If we were to reduce the number of dimensions to 1, then an immediate solution would be to project them on to a single axis and lose the other. However not all points may still retain their neighbors faithfully. Therefore a reasonable solution would be to consider the faithfulness of neighbors over a range of points in both spaces instead of all points. This gives us a smooth measure of precision and recall. Based on this idea, the **trustworthiness** and **continuity** measures were introduced [35]. Points are ranked in both the input space and the projection space based on their pairwise distances. Intuitively the ratio of match between the ranking from the projected space and that of the original space gives trustworthiness while continuity is measured in the opposite direction. Based on trustworthiness and continuity measures, a few extensions to point-wise quality measures were introduced in [18].

Some of the other evaluation methods of DR are namely visual assessment (as reported in [34]), reconstruction error [17] provided a reverse mapping is possible, classification error on projections [33] , preservation of data structures such as neighborhoods [34].

1.3 Highlights and Structure of the Thesis

Highlights of this thesis

We notice that DR methods in general are preferred as preprocessing techniques thereby expected to preserve the internal structure of the data. Some methods also do produce good visualizations that support empirical analyses.

Inspired by the idea of neighborhood retaining techniques we aim to focus on a supervised DR method that can retain neighbors in a stochastic manner. The advantage of using probabilities over distances is that widely separated points are considered equally farther and assigned similar memberships unlike distances. This allows focusing more on the closeby neighbors and wasting lesser probabilities on the wider ones. A common neighborhood function is used to provide this information during optimization. Therefore we formally define Supervised Probability Preserving Projection (SPPP) as a method that learns a linear transformation matrix that minimizes the probabilistic differences between the input covariates and their reponses given their neighborhood function. The probabilities in practice are obtained by normalizing the pairwise distances based on three different distributions namely Gaussian, Heavy-tail and Linear. All the three distributions share a common cost function but have different gradients. Experiments on both synthetic and real world data sets show that SPPP portrays equivalent or better performances compared to other recent state-of-the-art methods.

The contributions of this thesis begin by suggesting a probabilistic extension to a more recent technique called Supervised Distance Preserving Projections (SDPP). This is supported by synthetic examples for regression and classification tasks. Building on this idea the most significant contribution of this thesis is in introducing a novel supervised probabilistic DR technique dubbed as Supervised Probability Preserving Projection (SPPP). This includes formulation and optimization of SPPP followed by experiments and discussions on both synthetic and real world data sets.

Structure of the thesis

- Since this thesis builds on the work of SDPP, Chapter 2 briefly presents the formulation and optimization of SDPP and its neighborhood selection heuristics. This is followed by visualization on two synthetic examples.

- Chapter 3 suggests the notion of probabilistic neighborhoods for SDPP supported by visualizations on two synthetic examples.

- The main contribution of this thesis towards supervised DR is introduced in Chapter 4 called SPPP. The technique is discussed in detail beginning from the motivation, formulation and optimization, leading to experiments.

- Chapter 5 presents experimental verifications on real world data sets compared to other recent state-of-the-art methods for regression and classification tasks followed by discussions. Finally a summary of the work is provided.

Chapter 2

A brief look at SDPP

This chapter discusses a recently proposed supervised DR method called Supervised Distance Preserving Projections (SDPP) [39] that aims to preserve local information based on pairwise distances. SDPP finds a projection space where the differences between pairwise distances of the projected covariates and their responses are minimized locally.

2.1 Formulation

SDPP is motivated under the Weierstrass definition of continuity. According to the definition if two input covariates are close, then their responses should also be close. Let us consider a given set of data points $\{\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n\} \in \mathbb{R}^d$ with corresponding responses $\{\mathbf{y}_1, \mathbf{y}_2, ...\mathbf{y}_n\} \in \mathbb{R}^m$. Assuming there is exists a continuous mapping f: $\mathcal{X} \rightarrow \mathcal{Y}$ and that the input space is well sampled, mathematically the continuity of a function f at a point $\mathbf{x} \in \mathcal{X}$ and for every $\epsilon_\mathbf{y} > 0$ there exists an $\epsilon_\mathbf{x} > 0$ such that for all $\mathbf{x} \in \mathcal{X}$:

$$||\mathbf{x} - \mathbf{x}'|| < \epsilon_\mathbf{x} \Rightarrow |f(\mathbf{x}) - f(\mathbf{x}')| < \epsilon_\mathbf{y}$$

To achieve this goal SDPP learns a suitable linear transformation matrix $\mathbf{W}$ which projects the input points to a projection subspace that minimizes the differences between pairwise distances of the covariates and their responses. An effect of this leads to a projection where the local geometry of points mimics that of their responses.

The transformation for a point $\mathbf{x}$ in the subspace is obtained as:

$$\mathbf{z} = \mathbf{W}^T\mathbf{x}$$

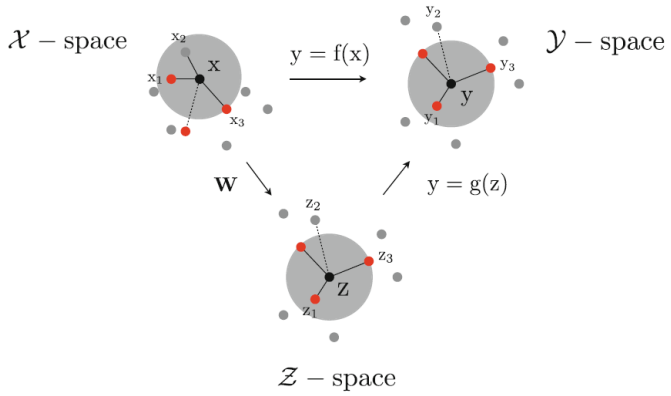


Figure 2.1: Illustration of SDPP

where $\mathbf{W} \in \mathbb{R}^{d*r}$ and r is the reduced dimensions.

The working principle of SDPP can be illustrated using figure 2.1. Let us consider a point $\mathbf{x}$ with nearest neighbors $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$. The transformation matrix $\mathbf{W}$ leads us to the $\mathcal{Z}$ space where the local geometry mimics that of the $\mathcal{Y}$ space. For this to happen, after projection, the point $\mathbf{x}_2$ (now $\mathbf{z}_2$) is moved outside the active neighborhood of point $\mathbf{x}$ (now $\mathbf{z}$) in the $\mathcal{Z}$ space while another point is moved inside the region.

The cost function of SDPP is given by:

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

where $N(\mathbf{x}_i)$ denotes the set of points that fall inside the neighborhood of the point $\mathbf{x}_i$, $d_{ij}^2(\mathbf{W}) = ||\mathbf{z}_i - \mathbf{z}_j||^2$ and $\delta_{ij}^2 = ||\mathbf{y}_i - \mathbf{y}_j||^2$. In a more compact form, the cost can be rewritten as,

$$\mathrm{J}(\mathbf{W}) = \frac{1}{n}\sum_{ij}\mathbf{G}_{ij}(\mathbf{D}_{ij} - \mathbf{\Delta}_{ij})^2 \tag{2.1}$$

$\mathbf{D}_{ij}$ is the distance between points i and j in the $\mathcal{Z}$ space and $\mathbf{\Delta}_{ij}$ is their distance in the $\mathcal{Y}$ space. $\mathbf{G}$ is the neighborhood matrix represented as follows,

$$\mathbf{G}_{ij} = \begin{cases} 1, & \text{if } \mathbf{x}_j \text{ is a neighbor of } \mathbf{x}_i \\ 0, & \text{otherwise.} \end{cases} \tag{2.2}$$

2.2 Optimization

The cost function of SDPP is quadratic, meaning twice differentiable at its minimum. This allows us to use the Conjugate Gradient (CG) optimization. CG is a classical numerical optimization technique that basically performs a line search in an iterative manner by finding the negative direction of the search vector and proceeding in that direction until the cost has been well minimized. Various methods have been proposed to speed up the gradient search, still CG may be unstable in cases where the search directions are not conjugate. Motivated by the fact that CG is still monotonic, preconditioning on the variables has been suggested by the Polak-Ribiére formula [21]. We use a minimization code following this method for all our experiments that uses CG. For a Semidefinite Quadratic Linear Programming (SQLP) based approach for optimization the interested reader is referred to [39].

Conjugate Gradient

The cost of SDPP mentioned above has a simple gradient when computed with respect to $\mathbf{W}$.

$$J(\mathbf{W}) = \frac{1}{n}\sum_{ij}\mathbf{G}_{ij}(\mathbf{D}_{ij} - \mathbf{\Delta}_{ij})^2 \tag{2.3}$$

$$\nabla_{\mathbf{W}} J = \frac{4}{n}\sum_{ij}\mathbf{G}_{ij}(\mathbf{D}_{ij} - \mathbf{\Delta}_{ij})\boldsymbol{\tau}_{ij}\boldsymbol{\tau}_{ij}^T\mathbf{W} \tag{2.4}$$

where $\boldsymbol{\tau}_{ij} = \mathbf{x}_i - \mathbf{x}_j$. A more compact form of (2.4) could be written after denoting $\mathbf{Q} = \mathbf{G} \odot (\mathbf{D} - \mathbf{\Delta})$ where $\odot$ represents the element-wise product of two matrices, the symmetric matrix $\mathbf{R} = \mathbf{Q} + \mathbf{Q}^T$ and a diagonal matrix $\mathbf{S} = \sum_j \mathbf{R}_{ij}$. The algebraic manipulations are shown below,

$$\begin{aligned}
\nabla_{\mathbf{W}} J &= \frac{4}{n}\sum_{ij}\mathbf{Q}_{ij}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{W} \\
&= \frac{4}{n}\sum_{ij}(\mathbf{x}_i\mathbf{Q}_{ij}\mathbf{x}_i^T + \mathbf{x}_j\mathbf{Q}_{ij}\mathbf{x}_j^T - \mathbf{x}_i\mathbf{Q}_{ij}\mathbf{x}_j^T - \mathbf{x}_j\mathbf{Q}_{ij}\mathbf{x}_i^T)\mathbf{W} \\
&= \frac{4}{n}\left[\sum_{ij}\mathbf{x}_i(\mathbf{Q}_{ij} + \mathbf{Q}_{ji})\mathbf{x}_i^T - \sum_{ij}\mathbf{x}_i(\mathbf{Q}_{ij} + \mathbf{Q}_{ji})\mathbf{x}_j^T\right]\mathbf{W}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{4}{n}\left(\sum_{ij}\mathbf{x}_i\mathbf{R}_{ij}\mathbf{x}_i^T - \sum_{ij}\mathbf{x}_i\mathbf{R}_{ij}\mathbf{x}_j^T\right)\mathbf{W} \\
&= \frac{4}{n}\left(\sum_{i}\mathbf{x}_i\sum_{j}\mathbf{R}_{ij}\mathbf{x}_j^T - \sum_{ij}\mathbf{x}_i\mathbf{R}_{ij}\mathbf{x}_j^T\right)\mathbf{W} \\
&= \frac{4}{n}\left(\sum_{i}\mathbf{x}_i\mathbf{S}_{ii}\mathbf{x}_j^T - \sum_{ij}\mathbf{x}_i\mathbf{R}_{ij}\mathbf{x}_j^T\right)\mathbf{W} \\
&= \frac{4}{n}(\mathbf{X}^T\mathbf{S}\mathbf{X} - \mathbf{X}^T\mathbf{R}\mathbf{X})\mathbf{W} \\
&= \frac{4}{n}\mathbf{X}^T(\mathbf{S}-\mathbf{R})\mathbf{X}\mathbf{W}
\end{aligned}
$$

where the rows in $\mathbf{X}$ denote input samples and ($\mathbf{S}$-$\mathbf{R}$) is the Laplacian matrix.

2.3 Neighborhood Selection in SDPP

The result of the optimization produces a transformation matrix $\mathbf{W}$ which may vary depending on the size of the neighborhood chosen before optimization. The neighborhood matrix given by $\mathbf{G}$ may be set using four different heuristics.

Not-Enclosing Neighborhoods

k-Nearest Neighbor

One of the simple heuristics for continuity preserving DR techniques is to choose a k-nearest neighborhood. This implies selecting k-nearest points and preserving their pairwise distances. Depending on the task, the value of k may then be learned using cross-validation. A heuristic popular among people in spectral clustering is to set k to be equal to $\log(n)$ [5]. Interestingly as the input sample size approaches infinity, $k$nn has been shown to achieve error rates no worse than twice the Bayes error rate [6] thereby displaying consistency.

ϵ -neighborhood

The ϵ-neighborhood on the other hand imposes a distance as a limit to picking the neighboring points. In other words points that lie within ϵ distance away from the point in question are selected as the nearest neighbors and their pairwise distances are preserved within the resulting neighborhood. A common heuristic used for

setting ϵ is usually $(log(n)/n)^{1/d}$ [20].

Enclosing Neighborhood

Enclosing k-Nearest Neighbor

Following the $k$nn approach a convex hull is constructed around each point with the smallest k such that those points that allow to completing the hull are included [9]. Those points that are not included within the so-constructed convex hull are left out of the neighborhood.

Natural Neighbors

Natural Neighbors are based on computing the Voronoi tessellation on the training points [27]. The natural neighbors of point i are those points whose cells are adjacent to that of i including i. The computational load of Voronoi tessellation is proportional to the number of points and the dimensionality. Clearly for very high dimensional cases this is an expensive choice.

The four methods discussed above are good starting points but still have certain shortcomings. $k$nn and ϵ-neighborhood are simple definitions made in the input space. But they are sensitive to local structure in the data especially affected even by small noise in the neighborhood. As for the enclosing neighborhood strategies, although they are geometrically balanced, the computational complexities in both cases increase exponentially with both number of samples and the dimensionality of the data set. The complexity involved in calculating the Voronoi tessellation is $O(nlog(n))$ when $d < 3$ and $O((n/2)^{d/2})$ when $d \geq 3$.

Continuity as a measure for neighborhood selection

Once the final projections have been obtained in general it is essential to estimate their quality compared to the original samples. Following our motivation we want to measure the continuity between $\mathcal{Z}$ space and $\mathcal{Y}$ space. Continuity as introduced in [35] is a measure of recall defined over a set of neighbors. Intuitively it can be understood as the match between the ranking of points on both spaces. Any new point intruding into a neighborhood in the projection will reduce the precision of the projection. Any point moving away will decrease its recall. In order to quantify how great the changes affect, it is important to know how far the points

have traveled to intrude or extrude the neighborhoods. We have that information in their pairwise distances in both spaces. Now simply ranking them for a range of points, gives us a means of quantifying their trustworthiness and continuity. Now let k_r be the range of the neighborhood provided by the user and $V_{kr}(i)$ be the set of points that are within the neighborhood of point i in the $\mathcal{Z}$ space but not in the $\mathcal{Y}$ space and let $r(i,j)$ be the rank of point $\mathbf{y}_j$ in the ordering based on its distance from point $\mathbf{y}_i$. Now the continuity is measured as,

$$M_{cont}(k_r) = 1 - C(k_r) \sum_{i=1}^{n} \sum_{j \in V_{k_r}(i)} (r(i,j) - k_r) \tag{2.5}$$

where the scaling term $C(k_r)$ is defined as,

$$C(k_r) = \begin{cases} \dfrac{2}{nk_r(2n - 3k_r - 1)}, & \text{if } k_r < \dfrac{n}{2} \\ \dfrac{2}{n(n-k_r)(n-k_r-1)}, & \text{if } k_r \geq \dfrac{n}{2} \end{cases} \tag{2.6}$$

The error term reaches its maximum when the compared ranking is completely reversed thereby making the continuity equal to 0. In all our experiments for regression we use this measure as an indicator value for choosing the right value of the parameter via cross-validation.

2.4 Synthetic examples

We now establish an experimental framework to illustrate the behavior of SDPP. We have two synthetic examples, one for regression and one for classification. The steps involved in handling the data set and the choices in parameter estimations are discussed below. This same framework will be used for illustration in future methods as well.

2.4.1 Data sets

"Parity" data

Lets us denote a set of 5 input variables by $\{X_1, X_2, ..X_5\}$ and a corresponding 1D response variable as Y. First we generate a set of 4000 input data points normally distributed in $[0,1]^5$. Then we use only the first and the second dimensions to compute the response variable as given below using a simple function,

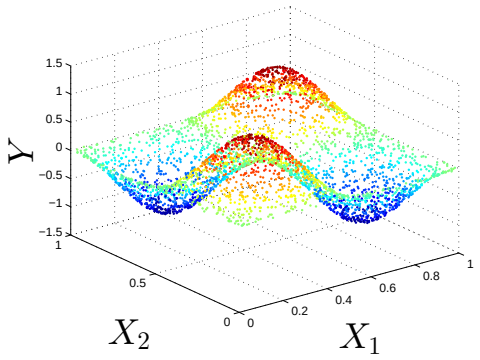


Figure 2.2: Parity

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

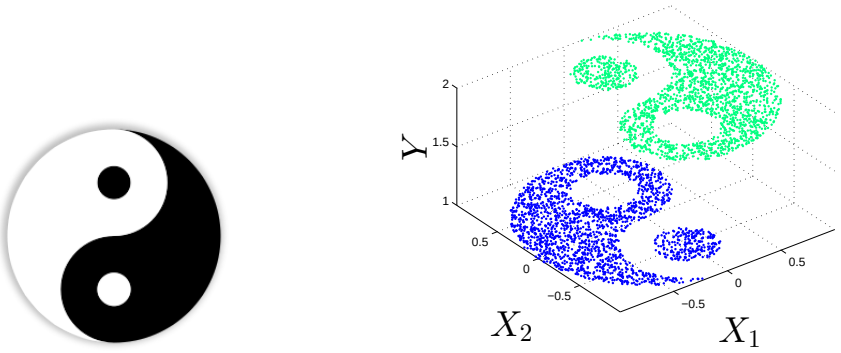
where ϵ is a noise term with $\epsilon \sim N(0, 0.1^2)$. Figure 2.2 shows the 3 dimensional (for clarity) depiction of the data. It is easy to see that much of the information could be preserved by simply 'flattening' the data into 2 dimensions.

Taijitu symbol

In order to observe the performance of SDPP for classification, we generate a data set resembling a symbol called 'Taijitu' (Figure 2.3(a)). This is an ancient Chinese symbol used often by Taoism representing opposite forces of harmony. The symbol is formed by dividing a circle into two halves with an S-curve and small dot-shaped areas of same class on opposite sides. To us this is interesting because of the disconnected nature of points belonging to the same class. The input data is generated by normally distributing a set of 4000 points with X_1, X_2 in $[0, 1]^2$ and X_3,X_4,X_5 as noise along $\epsilon \sim N(0, 0.1^2)$. The labels of points falling within a circle (based on X_1,X_2 alone) are assigned their class labels as $y = 1$ and $y = 2$ based on the region they occupy.

Experimental setup

As a preprocessing step, the data sets are first mean centered. Then they are divided into two parts with 80% for learning (sample size (n_l)) and the remaining 20% for testing (n_t). For regression, we learn the transformation matrix $\mathbf{W}$ on the training set and compute the continuity measure on the test projection from



(a) **Taijitu symbol** (b) **3D depiction**

Figure 2.3: Taijitu symbol data set

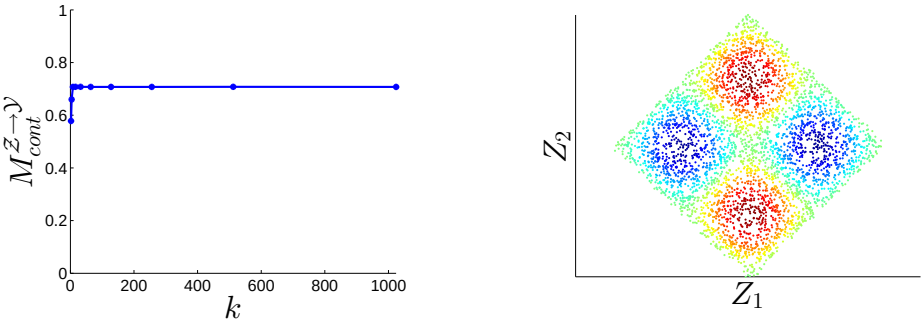
$\mathcal{Z} \rightarrow \mathcal{Y}$ space. We can denote this as $M_{cont}^{\mathcal{Z}\rightarrow\mathcal{Y}}$ for use within the plots. In practice the continuity is calculated for a set of neighborhood ranges $\mathbf{k}_r = \{k_{r1}, k_{r2}, ..k_{r5}\}$ uptil one half of the test set i.e. $k_{r5} \leq \frac{n_t}{2}$ so that we can avoid the influence of the scaling factor in the continuity definition. Thereby we get a vector with 5 continuity values $\mathbf{M}_{cont} = \{M_{cont1}, M_{cont2}, ..M_{cont5}\}$. Then we take the mean of these measured values to represent an estimate of the continuity of a method for a particular value of the parameter. The use of taking the mean is to indicate how consistent the parameter is in preserving local and a moderately wider neighborhood. So if we have 10 candidate parameters, we then have 10*5 continuity calculations, eventually reduced to 10*1 values which are also the ones plotted. For the classification task we use the classic k *nearest neighbor* (knn) classifier with k=1. Again the performance is calculated by learning a classifier on the projected training set and evaluating on the projected test set. Whichever candidate parameter obtains the best prediction accuracy is chosen for the final projections. For both regression and classification tasks, 2 dimensions is set as the desired reduced dimensions because the nature of the data in our synthetic cases are known beforehand. In reality, one could use cross-validation to find the best reduced dimensions.

There are very many different ways one could choose the candidate values for cross-validation. Specific to our current task, the range of parameters (here neighbors k) used for regression and classification are obtained by incrementing k in powers of 2 uptil around one half of the learning set has been considered as neighbors such as $\{2, 4, 8, 16, .. \leq \frac{nl}{2}\}$. In case the next value of k is just a little above $\frac{nl}{2}$, we also consider that k. Finally the parameter that obtains the best value of continuity for regression and accuracy for classification respectively are considered

for the final projection.

2.4.2 Regression: Parity data

Now we conduct a cross-validation on the parity data set to observe the performance of SDPP for regression.



(a) **CV** (b) **SDPP**

Figure 2.4: SDPP for regression

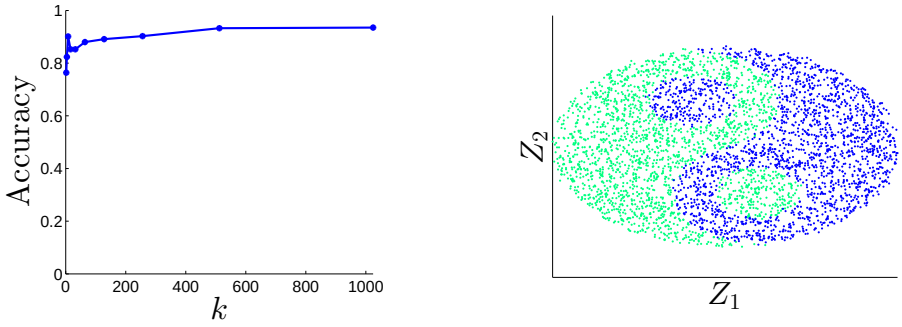
From the cross-validation chart in figure 2.4(a), we are able to choose from eight values of neighborhood sizes (k). Alternatively all values of neighborhoods within that range would perform similarly. After dimensionality reduction figure 2.4(b) shows that SDPP preserves the expected behavior by correctly finding the first two dimensions to be the significant dimensions and thereby producing a pleasing visualization. As SDPP tries to preserve its neighbors instructed by the neighborhood matrix $\mathbf{G}$, it is able to retain a similar structure on the reduced dimensions as well.

2.4.3 Classification: Taijitu symbol

SDPP naturally can be used for classification by simply encoding the class information in a manner that allows all classes to be equally spaced apart from each other. This ensures that the arbitrary nature of class labeling does not influence the pairwise distances of points in the output space. One simple way to encode class information is by setting class 1 as {0,1}, class 2 as {1,0} [1].

[1] Although in this case we only have two classes, so it suffices to leave them as they are.

The cross-validation chart in figure 2.5(a) shows that SDPP consistently performs well over a range of neighborhood sizes. The 2D projection as in figure 2.5(b) closely resembles the original symbol although with a slight skew.



(a) **CV** (b) **SDPP**

Figure 2.5: SDPP for classification

Chapter 3

Probabilistic Neighborhoods for SDPP

In this chapter we suggest the notion of probabilistic neighborhoods for SDPP instead of binary valued memberships as illustrated using two simple examples.

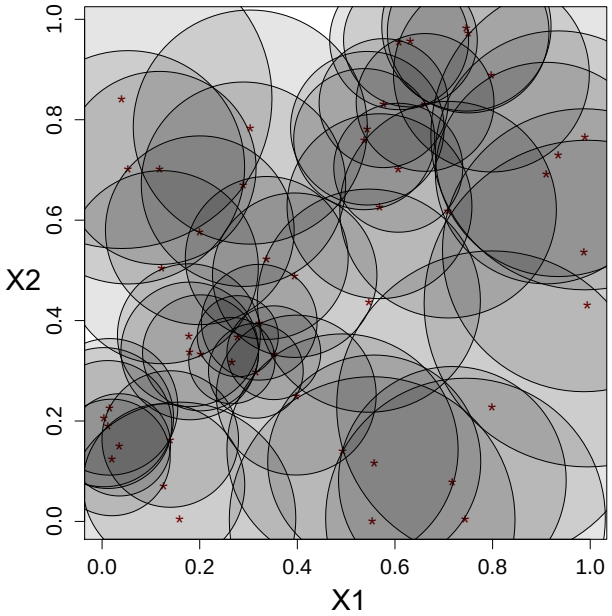


Figure 3.1: Impact of binary valued memberships with a fixed k for all points.

Imagine a space with a uniform distribution of data points as shown in figure 3.1 (X1,X2 are random variables). We see that for a fixed value of neighborhood

size $k = 5$ some areas are heavily covered and some, weakly. By setting binary valued memberships we are including all neighbors (inside the neighborhood) to have equal influence on a point. This has a greater impact when some data points are included in the cost more times than some other points simply due to a poor choice of k rather than their positioning in that space. This is worrying since the ground of dimensionality reduction is based on the assumption that closeby covariates exist because their stimuli must have been similar. In order to mitigate this problem we consider the influence of neighbors with relative strength. In other words we choose to assign probabilistic memberships for each point as explained below.

In our problem, we cast the points on to a distribution such as the Gaussian wherein the conditional probability $p_{j|i}$ of a point j to be a neighbor of point i varies depending on its distance from i as given by equation 3.1. Now nearby points will receive a relatively higher $p_{j|i}$ and the farther points will receive rather small membership values.

$$p_{j|i} = \frac{\exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||\mathbf{x}_i - \mathbf{x}_k||^2/2\sigma_i^2)} \tag{3.1}$$

where $||\mathbf{x}_i - \mathbf{x}_j||^2$ is the pairwise (Euclidean) distances between points i and j in the $\mathcal{X}$ space and σ_i is the variance of a Gaussian centered around point i. Because we are only interested in modeling the pairwise similarities we set $p_{i|i} = 0$. The probabilities are dependent on two variables: the pairwise distances and the variance (σ) of the Gaussian. The pairwise distances are fixed once the data set is compiled. But the σ is a free parameter which can be used to control the probabilities assigned. A carefully chosen value of σ allows us to utilize the probabilities to focus on the real neighbors and not waste them on farther points, meaning the wider points are all considered equally wide beyond a limit.

But how do we set this value for σ? Would it suffice to fix a single value for all points? It is not likely that every point has a similar density of points surrounding them. Focusing on a proportion of neighbors instead of all is vital in some cases. To illustrate this, let us consider another example as shown in figure 3.2 where the neighbors (x-axis) are plotted against distance (y-axis) for a set of 300 points normally distributed. It is easy to see that from a certain point i the distance to reaching the nearest 150 points (half the size of samples) could vary anywhere approximately between 0.18 to 0.62 units. This will have a greater impact when the input distribution is skewed in practice thereby wasting some probabilities on outliers; implying that not all points share the same value of variance. Therefore

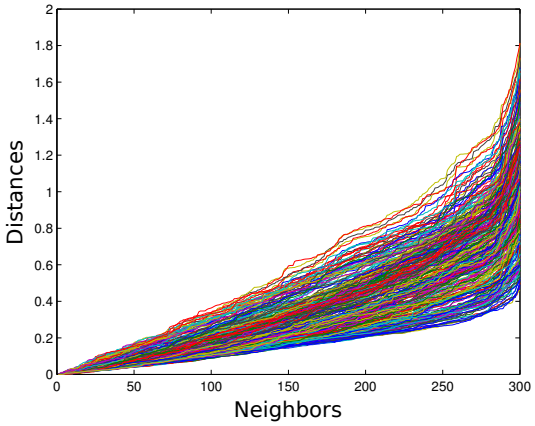


Figure 3.2: A plot illustrating the differences in variances while reaching half the sample size from some point i.

we smoothen the reach of points so that all points get their neighborhood weighting value as a function of their density controlled by the variance. In other words the membership values of points around i 'fade' with a steep or a wide gaussian based on the neighborhood density. For this reason it is essential to compute unique σ_i for each point.

3.1 Entropy and Cross entropy for setting variance in neighborhoods

In order to obtain a fair value for σ_i for each point, we seek an information theoretic approach of using *entropy* as a measure of information [26]. In our probability function, any value of σ_i induces a $\mathbf{p}_i$ over all other points. This probability distribution has an entropy that is monotonic with the value of σ_i. Hence a binary search can be performed for a value of σ_i that produces a $\mathbf{p}_i$ with an entropy that matches the perplexity provided by the user. The perplexity is now our probabilistic equivalent of k. The perplexity is measured in bits and may be understood as a smooth measure of effective number of neighbors which is computed as,

$$Perp(\mathbf{p}_i) = 2^{H(\mathbf{p}_i)} \tag{3.2}$$

where $H(\mathbf{p}_i)$ is the Shannon entropy of $\mathbf{p}_i$ calculated as

$$H(\mathbf{p}_i) = -\sum_j p_{j|i} \log_2 p_{j|i} \tag{3.3}$$

Using this approach, we obtain a neighborhood matrix representing memberships of points according to the perplexity provided by the user. Unlike the neighborhood matrix used in the previous chapter, this one aims to indicate the stochastic relations between points allowing us to relatively focus more on some points and less on some others.

While using *entropy* serves the purpose in most cases, we consider the use of *cross entropy* as well. Cross entropy between two distributions measures the average number of bits required to encode a certain information if the encoding scheme was based on an alternative distribution rather than the true distribution. This implies that in our case we use the distribution of the response variable instead to approximate that of the input variables. The idea behind this approach is that during cases where the input samples are not very informative by themselves or are corrupted by noise, the outputs (responses) might give the necessary supplementary information in the form of subtle supervision. On the other hand when the data violates the surjective (one to many or vice versa relations) definition, this could induce complexities by itself. Hence knowledge about the data beforehand is helpful, although experimental verifications could assist in making final decisions. The conditional probabilities in $\mathcal{Y}$ space are given by equation 3.4

$$q_{j|i} = \frac{\exp(-||\mathbf{y}_i - \mathbf{y}_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||\mathbf{y}_i - \mathbf{y}_k||^2/2\sigma_i^2)} \tag{3.4}$$

The cross entropy is then expressed as,

$$H(\mathbf{p}_i, \mathbf{q}_i) = -\sum_j p_{j|i} \log_2 q_{j|i} \tag{3.5}$$

As given by the equation 3.5, the appropriate values of σ_i, common to both spaces, that produce a cross entropy to match the perplexity given by the user are calculated. SNE [12] and t-SNE [32] compute entropy within their implementations which we adopt with necessary modifications to also compute cross entropy in our experiments.

3.2 Student's t-distribution for neighborhoods

Entropy and Cross entropy based neighborhoods are still parametric methods where the value of σ indirectly needs to be found using cross-validation for perplexity. However by casting the data points on to some other non-parametric distribution such as the **t-distribution**, we obtain another neighborhood function. The advantage here lies in avoiding cross-validation, though the downside in this approach is that it cannot be fine-tuned for improving the performance. The input conditional probabilities for t-distribution with one degree of freedom is given by 3.6. For our experiments, we only consider the input space to compute the neighborhood matrix since the output space alone could mislead us with no consent of the inputs.

$$p_{j|i} = \frac{(1 + ||\mathbf{x}_i - \mathbf{x}_j||^2)^{-1}}{\sum_{k \neq i}(1 + ||\mathbf{x}_i - \mathbf{x}_k||^2)^{-1}} \tag{3.6}$$

3.3 Synthetic examples

The experimental setup remains the same as before, but now the parameter cross-validated for is perplexity ($Perp$) instead of the number of neighbors. We again have 10 candidate parameter values for perplexity incremented in powers of 2 as explained before. After all the perplexity is simply a probabilistic equivalent of k. Further we would like to compare their performance for a similarly varying parameter choices.

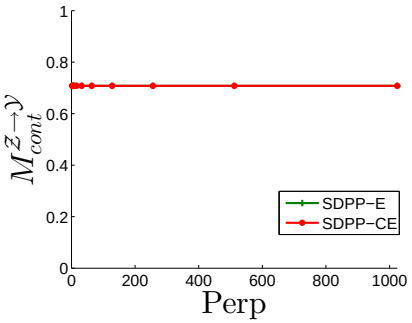
3.3.1 Regression: Parity data

While using the same formulation of SDPP and merely applying a new neighborhood function using entropy and cross entropy, we see that the structure is not only preserved (as in fig 3.3(b) and fig 3.3(c)) but also portrays consistent performance over all the values of perplexity unlike standard SDPP. This reinstates the fact that smooth memberships for neighbors are in cases better than discete binary valued memberships. Another interesting observation shows that the cross entropy based neighborhood is equally as good as entropy for this data set. The student based neighborhood however does not preserve the structure (fig 3.3(d)). This is partly because the points falling at the boundaries of the contour (green or yellow) are around regions of uncertainty in addition to having more uncertain

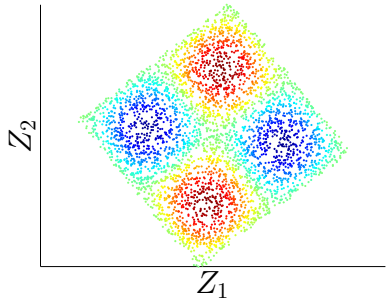
points near them. But the points lying in the peaks of the contours (red or blue) are more likely to be of the same nature. Unfortunately the non-parametric nature of student distribution does not let us limit the number of influencing neighbors, thereby resulting in a less pleasing visualization. Although unfair that PCA being an unsupervised method, it is interesting to compare the projection for a similar reason as in student. It is easy to see that PCA mixes up points by virtue of equivalence in their variances along the first two principal axes which is relative to pairwise distances in our case.

3.3.2 Classification: Taijitu Symbol

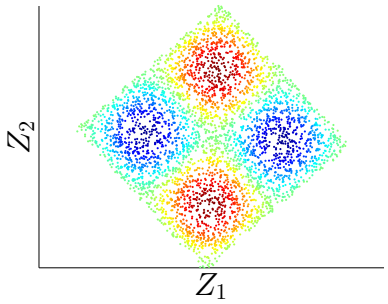
Using the same **Taijitu** data set and experimental setup, we illustrate the performance of probabilistic neighborhood selection for classification. Figures 3.4(b) and 3.4(c) show that both entropy and cross entropy based neighborhoods can maintain the structure of the symbol although with a similar skew like the standard SDPP. In this case, the student neighborhood also seems to provide a good visualization by preserving the original structure(fig 3.4(d)). The fact that points belonging to opposite classes are equally apart (thanks to the class labels), allows student distribution to separate them without the confusion induced by the tail of the distribution. Again for the same reason we look at PCA on the other hand completely mixing points from both classes once again due to relatively equal variances in both the principal axes.



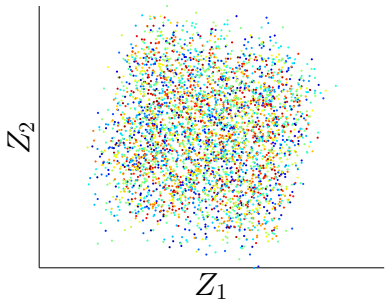
(a) **CV**



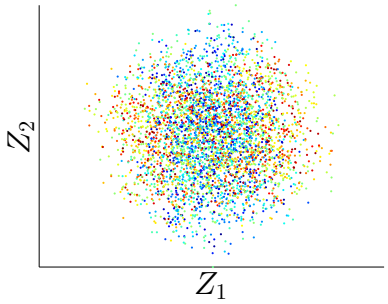
(b) **SDPP-entropy**



(c) **SDPP-cross entropy**

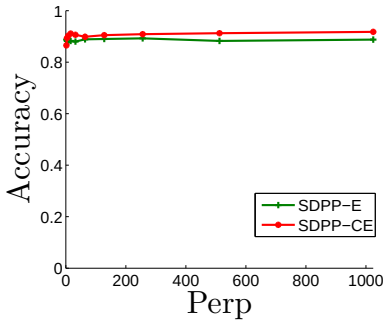


(d) **SDPP-student**

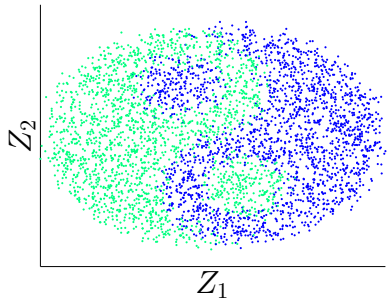


(e) **PCA**

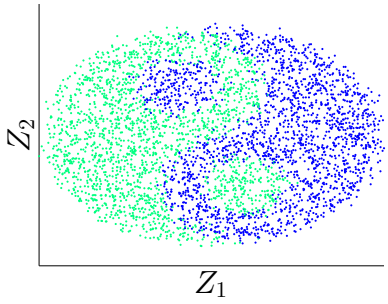
Figure 3.3: SDPP with probabilistic neighborhoods for regression



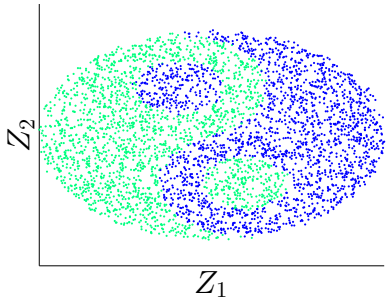
(a) **CV**



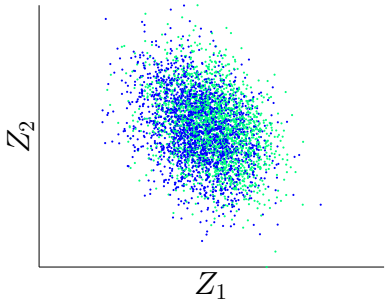
(b) **SDPP-entropy**



(c) **SDPP-cross entropy**



(d) **SDPP-student**



(e) **PCA**

Figure 3.4: SDPP with probabilistic neighborhoods for classification

Chapter 4

Supervised Probability Preserving Projection (SPPP)

In this chapter we introduce a fully probabilistic supervised DR technique called Supervised Probability Preserving Projection (SPPP). As the name suggests, here we aim to preserve the probabilistic relations between points by finding a projection space where the probabilistic dissimilarities between the input covariates and their responses are minimized. We note that in most cases DR techniques are employed as preprocessing tools, meaning they are expected to retain the intrinsic structure of the original data. Some other techniques are useful for producing good visualizations. We show that SPPP could be useful in both cases, later with experiments.

4.1 Motivation

In the previous chapter we noticed that using a probabilistic neighborhood function allowed us to utilize the probabilities in a controlled manner to improve the performance. However still the original differences in the cost function of SDPP operate on pairwise distances. Instead, by replacing them with probabilities, we aim to worry less about wider points unlike distances, where how wide a point lies is still added into the cost. With this as the motivation, we aim to define a fully probabilistic method.

The Weierstrass definition of continuity used in SDPP is typically defined for distances, hence they do not implicitly hold for probabilities. However the principle of Weighted Individual Differences model provides a platform for us to formulate a similar looking cost function as in SDPP, but only this time in a probabilistic

setting.

The Weighted Individual Differences model tries to minimize the square of the differences between two scalar metrics given their weight [28]. Although probabilities are not typically metrics, they are still monotonic in nature, since they are based on pairwise distances. The farther the points are, the lesser are their probabilities of being neighbors. By defining an alternative function such as the plain Linear function, which we will see later, we still obtain a monotonic varying function that could be considered a metric. In the Linear function, the farther points are more likely to **not** being neighbors than the nearby points. A simple case of the Weighted individual differences model can be expressed as,

$$\mathbf{Diff}_{ij} = \mathbf{Weight}_{ij}(\mathbf{E}_{ij} - \mathbf{F}_{ij})^2$$

where **Weight** is the weighting matrix and $\mathbf{E}$ and $\mathbf{F}$ are some arbitrary matrices containing metric values whose differences are to be minimized. All the matrices shown above are equi-dimensional where scalar $\mathbf{Diff}_{ij}$ represents the weighted difference between the i_{th} row and j_{th} column elements of $\mathbf{E}$ and $\mathbf{F}$. When all $\mathbf{Weight}_{ij} = 1$, it is called unweighted individual differences. Before defining the cost function, it would be useful to illustrate the working principle of SPPP.

Illustration

Let us consider a given set of data points $\{\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n\} \in \mathbb{R}^d$ with corresponding responses $\{\mathbf{y}_1, \mathbf{y}_2, ...\mathbf{y}_n\} \in \mathbb{R}^m$.

The aim here is to obtain a linear transformation matrix $\mathbf{W}$ that minimizes the differences between the probabilistic similarities of the projected covariates and their responses. A point $\mathbf{x}$ in the projection space is computed as $\mathbf{z} = \mathbf{W}^T\mathbf{x}$ where the transformation matrix is $\mathbf{W} \in \mathbb{R}^{d*r}$, r being the reduced dimensions. We believe that a completed optimization preserves the probabilistic arrangement in the $\mathcal{Z}$ space that closely resembles that of the $\mathcal{Y}$ space.

SPPP can be illustrated using the figure 4.1. Let us consider the central point in *black* to be of interest. The pairwise similarities between itself and all its neighbors are indicated by the rainbow color scale with black color indicating 0.[1] Starting

[1]The color scale represents the notion of memberships analogous to the wavelength of the colors ranging from red with the highest and violet with the lowest. The color scale may be reversed for the Linear function.

at the $\mathcal{X}$ space, a linear subspace mapping is performed with $\mathbf{W}$ leading to the $\mathcal{Z}$ space. Here we notice that the two *orange* points are moved slightly apart from each other matching their pairwise similarities in the $\mathcal{Y}$ space. Similarly the points in *green* (both light and dark) are only closer to one of the *orange* points unlike in the input space. The point in *blue* also experiences a small perturbation with the influence from other pairwise similarities. Notice that all these points are moved only along their own orbit which is an indication of maintaining their consistent pairwise similarities from the central point (*black*). Moreover the closer points are actively moved along their orbit while the wider points are relatively coarsely moved.

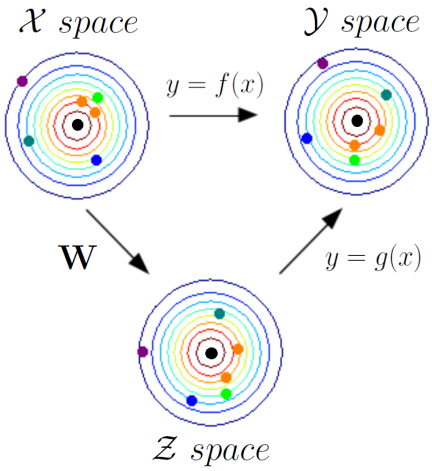


Figure 4.1: Probabilistic arrangement of neighbors illustrating the objective of SPPP.

4.2 Cost function

Based on the above motivation and illustration, we formulate our cost function. First we convert the pairwise distances into probabilities by assuming the data to follow one of the common distributions such as the Gaussian, Heavy-tail or Linear. Then we can solve for $\mathbf{W}$ as the matrix that minimizes the cost. The cost function is quadratic allowing us to make use of the conjugate gradient optimization.

$$C(\mathbf{W}) = \frac{1}{n} \sum_{ij} \mathbf{G}_{ij} (\log \mathbf{P}_{ij} - \log \mathbf{Q}_{ij})^2 \tag{4.1}$$

where **P** and **Q** are now the joint probability distributions in the projection and response spaces repectively. Computing these matrices differ based on the function chosen, hence we will discuss them separately in the next section. **G** is a probabilistic neighborhood matrix that can be computed using one of the three methods described in chapter 3. So we can defer this choice until the experiments.

Each one of the selected functions for computing **P** and **Q** involves deriving a specific gradient of its own. Two recent methods SNE and t-SNE have pointed out that using Gaussian and Heavy-tail distributions are useful for providing good representations and mitigating the 'crowding problem' respectively in addition to obtaining simple gradients.[2] In addition to this we also consider the classic case of a linear distribution where no assumption is made about the data and is allowed to compute probabilistic similarities as defined by the nature of the data. Finally we formulate the three cases below as a problem of finding the projection matrix which will also allow us to accommodate out-of-sample points without having to run the optimization again.

The next section discusses the three versions of SPPP individually. Following that we present a brief set of differences and similarities between SPPP and two other recent probabilistic embedding methods, SNE and t-SNE. Finally we show the visualizations obtained on two synthetic examples, one in each of regression and classification.

4.3 Gradients

In order to compute the probabilities, we require the pairwise distances of points both in the projection space and in the response space. The pairwise distances are computed as,

$$dz_{ij}^2 = \boldsymbol{\tau}_{ij}^T \mathbf{W}\mathbf{W}^T \boldsymbol{\tau}_{ij} \quad , \quad dy_{ij}^2 = ||\mathbf{y}_i - \mathbf{y}_j||^2$$

where $\boldsymbol{\tau}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ for simplicity and the distances are typically Euclidean.

4.3.1 Gaussian

Now the conditional probability of a point j to be a neighbor of point i in the projection space is given by,

[2] A heavy-tail distribution with infinite degrees of freedom tends to a Gaussian

$$p_{j|i} = \frac{\exp(-dz_{ij}^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-dz_{ik}^2/2\sigma_i^2)} \tag{4.2}$$

Similarly the conditional probability of a point j to be a neighbor of point i in the response space is given by,

$$q_{j|i} = \frac{\exp(-dy_{ij}^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-dy_{ik}^2/2\sigma_i^2)} \tag{4.3}$$

But we are only interested in the joint probabilities such that

$$\mathbf{P}_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \quad , \mathbf{Q}_{ij} = \frac{q_{j|i} + q_{i|j}}{2n}$$

This is due to two reasons. Firstly it ensures that the probabilities are symmetric and monotonic and secondly it simplifies the derivation. Hence it is ensured that both joint distributions individually sum up to 1 while self similarities are set to 0 (such as $\mathbf{P}_{ii} = 0$ and $\mathbf{Q}_{ii} = 0$). The information about the influence of neighbors on individual points is provided by our common neighborhood (weighting) function $\mathbf{G}$. Hence it suffices to set the σ_is within both $\mathbf{P}$ and $\mathbf{Q}$ to a constant $\frac{1}{\sqrt{2}}$ making the term $2\sigma_i^2$ equal to 1. This means that all points spread a similar Gaussian assigning memberships as appropriate within their spaces. Finally this leaves us with the joint probabilities as in equation 4.4 with the denominators of the right side of both expressions individually summing to 1.

$$\mathbf{P}_{ij} = \exp(-dz_{ij}^2) \quad , \quad \mathbf{Q}_{ij} = \exp(-dy_{ij}^2) \tag{4.4}$$

Now we can find the gradient of the cost with respect to $\mathbf{W}$

$$C(\mathbf{W}) = \frac{1}{n}\sum_{ij}\mathbf{G}_{ij}(\log\mathbf{P}_{ij} - \log\mathbf{Q}_{ij})^2 \tag{4.5}$$

Since $\mathbf{W}$ resides within $\mathbf{P}$, we can use the Chain rule of differentiation to obtain the gradient. If we denote $\mathbf{Dz}^2$ as a matrix containing scalars dz_{ij}^2 for clarity,

$$\frac{\partial C}{\partial \mathbf{W}} = \frac{\partial C}{\partial \mathbf{Dz}^2}\frac{\partial \mathbf{Dz}^2}{\partial \mathbf{W}} \tag{4.6}$$

Starting with the second part of the chain rule,

$$dz_{ij}^2 = \boldsymbol{\tau}_{ij}^T \mathbf{W}\mathbf{W}^T \boldsymbol{\tau}_{ij}$$

$$\frac{\partial dz_{ij}^2}{\partial \mathbf{W}} = \boldsymbol{\tau}_{ij}\boldsymbol{\tau}_{ij}^T \mathbf{W} \tag{4.7}$$

Now handling the first part of the chain rule. Since any change in $\mathbf{dz}_i^2$ only changes dz_{ij}^2 and dz_{ji}^2 $\forall j$,

$$\frac{\partial C}{\partial \mathbf{dz}_i^2} = \sum_j \left(\frac{\partial C}{\partial dz_{ij}^2} + \frac{\partial C}{\partial dz_{ji}^2}\right) = 2\frac{\partial C}{\partial dz_{ij}^2} \tag{4.8}$$

Recall that $\mathbf{P}_{ij} = \exp(-dz_{ij}^2)$. Consequently,

$$\frac{\partial C}{\partial \mathbf{dz}_i^2} = \frac{\partial}{\partial \mathbf{dz}_i^2}\left\{\frac{1}{n}\sum_j \mathbf{G}_{ij}(\log(\exp(-dz_{ij}^2)) - \log \mathbf{Q}_{ij})^2\right\}$$

$$= \frac{2}{n}\sum_j \mathbf{G}_{ij}(-dz_{ij}^2 - \log \mathbf{Q}_{ij})(-1) \tag{4.9}$$

Now bringing together the three parts from equations 4.7, 4.8 and 4.9 , we have,

$$\frac{\partial C}{\partial \mathbf{W}} = \sum_i \frac{4}{n}\sum_j \mathbf{G}_{ij}(\log \mathbf{P}_{ij} - \log \mathbf{Q}_{ij})\boldsymbol{\tau}_{ij}\boldsymbol{\tau}_{ij}^T \mathbf{W}$$

$$\nabla_{\mathbf{W}} C = \frac{4}{n}\sum_{ij} \mathbf{G}_{ij}(\log \mathbf{P}_{ij} - \log \mathbf{Q}_{ij})\boldsymbol{\tau}_{ij}\boldsymbol{\tau}_{ij}^T \mathbf{W} \tag{4.10}$$

Interestingly this resembles the gradient of standard SDPP except for the probabilities. Hence the gradient can be simplified with minor algebraic manipulations into a more compact form. Denote $\mathbf{M} = \mathbf{G} \odot (\log \mathbf{P} - \log \mathbf{Q})$, $\mathbf{R} = \mathbf{M} + \mathbf{M}^T$ and $\mathbf{S}$ is a diagonal matrix with $\mathbf{S}_{ii} = \sum_j \mathbf{R}_{ij}$ as shown for SDPP.

$$\nabla_{\mathbf{W}} C = \frac{4}{n}\mathbf{X}^T(\mathbf{S} - \mathbf{R})\mathbf{X}\mathbf{W} \tag{4.11}$$

4.3.2 Heavy-tail

We use a heavy-tail more specifically a Student's t-distribution in this method. The probability density function of a t-distribution is given by,

$$pdf_t = c(1 + \frac{x^2}{v})^{-\frac{v+1}{2}}$$

where x^2 are pairwise distances. Setting v as 1 (one degree of freedom), gives us the following form (ignoring constant c)

$$pdf_t = (1 + x^2)^{-1}$$

Using the same notation from Gaussian, we obtain the conditional probabilities in the projection space as,

$$p_{j|i} = \frac{(1 + dz_{ij}^2)^{-1}}{\sum_{k \neq i}(1 + dz_{ik}^2)^{-1}} \tag{4.12}$$

and the output conditional probabilities are given by,

$$q_{j|i} = \frac{(1 + dy_{ij}^2)^{-1}}{\sum_{k \neq i}(1 + dy_{ik}^2)^{-1}} \tag{4.13}$$

Then we compute the joint probabilities for same reasons mentioned earlier. We ensure that the joint probability matrices individually sum to 1 and their self probabilities are set to 0 ($\mathbf{P}_{ii} = 0$ and $\mathbf{Q}_{ii} = 0$). Finally we get,

$$\mathbf{P}_{ij} = (1 + dz_{ij}^2)^{-1} \quad , \quad \mathbf{Q}_{ij} = (1 + dy_{ij}^2)^{-1} \tag{4.14}$$

The cost is again written below for convenience.

$$C(\mathbf{W}) = \frac{1}{n}\sum_{ij} \mathbf{G}_{ij}(\log \mathbf{P}_{ij} - \log \mathbf{Q}_{ij})^2 \tag{4.15}$$

Using the Chain rule from 4.6 to simplify the derivation.

$$dz_{ij}^2 = \boldsymbol{\tau}_{ij}^T \mathbf{W}\mathbf{W}^T \boldsymbol{\tau}_{ij} \qquad ; \qquad \frac{\partial dz_{ij}^2}{\partial \mathbf{W}} = \boldsymbol{\tau}_{ij}\boldsymbol{\tau}_{ij}^T \mathbf{W} \tag{4.16}$$

Under the same framework as before, we have

$$\frac{\partial C}{\partial \mathbf{dz}_i^2} = 2\frac{\partial C}{\partial dz_{ij}^2} \tag{4.17}$$

since any change in $\mathbf{dz}_i^2$ only changes dz_{ij}^2 and dz_{ji}^2 $\forall j$. However the second part undergoes a minor change.

Recall that $\mathbf{P}_{ij} = (1 + dz_{ij}^2)^{-1}$,

$$\frac{\partial C}{\partial \mathbf{dz}_i^2} = \frac{2}{n} \sum_j \mathbf{G}_{ij} (\log(1 + dz_{ij}^2)^{-1} - \log \mathbf{Q}_{ij})(1 + dz_{ij}^2)(-1)(1 + dz_{ij}^2)^{-2} \tag{4.18}$$

Combining the three parts in 4.16, 4.17 and 4.18 we have,

$$\nabla_{\mathbf{W}} C = \frac{4}{n} \sum_{ij} \mathbf{G}_{ij} (\log \mathbf{P}_{ij} - \log \mathbf{Q}_{ij}) \ (-1)(1 + dz_{ij}^2)^{-1} \ \boldsymbol{\tau}_{ij} \boldsymbol{\tau}_{ij}^T \mathbf{W} \tag{4.19}$$

Notice that $-(1 + dz_{ij}^2)^{-1}$ is now a scaling factor in that it only affects in how sooner or later we arrive at the optimum. Besides that the gradient can still be simplified with minor algebraic manipulations into a more compact form like before. Denote $\mathbf{M} = \mathbf{G} \odot (\log \mathbf{P} - \log \mathbf{Q})$, $\mathbf{R} = \mathbf{M} + \mathbf{M}^T$ and $\mathbf{S}$ is a diagonal matrix with $\mathbf{S}_{ii} = \sum_j \mathbf{R}_{ij}$.

$$\nabla_{\mathbf{W}} C = \frac{4}{n} \mathbf{X}^T (\mathbf{S} - \mathbf{R}) \mathbf{X} \mathbf{W} \tag{4.20}$$

4.3.3 Linear

In the case of a linear distribution, we normalize the pairwise distances in order to get the probabilities. Note that this time the probabilities indicate the likelihood of **not** being neighbors, while still being monotonic. The farther a point is, higher the probability of it not being a neighbor. The closer it is, lesser the probability of it not being a neighbor. Intuitively when all points try to push their corresponding wider points by a certain amount, it results in a probabilistic arrangement because a point that is closer to some point is always wider to some other point. Fortunately we still have the supervision to indicate how much they match the arrangement in the $\mathcal{Y}$ space. Again the conditional probability in the projection space for a point j to not be a neighbor of point i is given by,

$$p_{j|i} = \frac{dz_{ij}^2}{\sum_{k \neq i} dz_{ik}^2} \tag{4.21}$$

similarly the corresponding output conditional probabilities are,

$$q_{j|i} = \frac{dy_{ij}^2}{\sum\limits_{k \neq i} dy_{ik}^2} \tag{4.22}$$

The joint probabilities are computed again such that they individually sum up to 1 and the self probabilities are set to 0 ($\mathbf{P}_{ii} = 0$ and $\mathbf{Q}_{ii} = 0$). Finally we have,

$$\mathbf{P}_{ij} = dz_{ij}^2 \quad , \quad \mathbf{Q}_{ij} = dy_{ij}^2 \tag{4.23}$$

The cost is once again written below for convenience.

$$C(\mathbf{W}) = \frac{1}{n} \sum_{ij} \mathbf{G}_{ij} (\log \mathbf{P}_{ij} - \log \mathbf{Q}_{ij})^2 \tag{4.24}$$

By following the chain rule approach, we notice that the linear function has a simpler gradient than heavy-tail, which can be written in a more condensed form as shown in the previous cases.

$$\nabla_{\mathbf{W}} C = \frac{4}{n} \sum_{ij} \mathbf{G}_{ij} (\log \mathbf{P}_{ij} - \log \mathbf{Q}_{ij}) \ (1/dz_{ij}^2) \ \boldsymbol{\tau}_{ij} \boldsymbol{\tau}_{ij}^T \mathbf{W} \tag{4.25}$$

Consequently the gradient can be simplified with minor algebraic manipulations into a more compact form. Denote $\mathbf{M} = \mathbf{G} \odot (\log \mathbf{P} - \log \mathbf{Q})$, $\mathbf{R} = \mathbf{M} + \mathbf{M}^T$ and $\mathbf{S}$ is a diagonal matrix with $\mathbf{S}_{ii} = \sum\limits_j \mathbf{R}_{ij}$.[3]

$$\nabla_{\mathbf{W}} C = \frac{4}{n} \mathbf{X}^T (\mathbf{S} - \mathbf{R}) \mathbf{X} \mathbf{W} \tag{4.26}$$

Differences and Similarities between SPPP and SNE/t-SNE

Although SPPP may appear similar to SNE and t-SNE in using probabilities, the following points are useful in order to appreciate the method.

Differences between SPPP and SNE/t-SNE

- SPPP is a supervised technique while SNE and t-SNE are unsupervised.
- SPPP optimizes for a transformation matrix $\mathbf{W}$ for also projecting out-of-sample points unlike the others that optimize only for a projection space.
- SPPP is aimed to operate as a DR technique, SNE/t-SNE are motivated for visualization (dim=2,3).

[3] $(1/dz_{ij}^2)$ is again a scaling factor within $\mathbf{M}$.

- SPPP cost and gradients are slightly different from that of SNE and t-SNE.
- SPPP may also use cross entropy to compute the neighborhood matrix, the other techniques use only entropy for their input space distributions (but do not explicitly have a neighborhood matrix)
- SPPP minimizes the weighted individual differences while the other minimize the KL divergence.

Similarities between SPPP and SNE/t-SNE

- All the three techniques use probabilities in their computations instead of plain pairwise distances.
- SNE uses Gaussian distribution on both spaces and SPPP uses this as one option.
- t-SNE uses t-distribution in the output and Gaussian in the input space while SPPP uses t-distribution in both the output space and the projection space (not the input space) as one option.

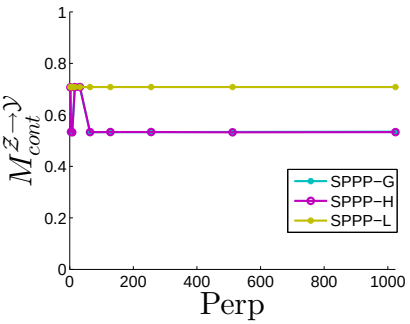
4.4 Synthetic examples

Once again we make use of the same two synthetic examples for regression (Parity) and classification (Taijitu). The experimental setup follows as explained for standard SDPP and the parameter selection follows from the previous chapter. This time we cross-validate for the correct value of perplexity used in computing the neighborhood matrix $\mathbf{G}$ and not that within $\mathbf{P}$ and $\mathbf{Q}$ because we set those values to be a constant $\frac{1}{\sqrt{2}}$ (for Gaussian). The neighborhood matrix $\mathbf{G}$ can be computed either using entropy or cross entropy. As a general heuristic, we chose entropy if the PCA projections obtained good performance (measure depends on the task), otherwise cross entropy. This is because PCA is unsupervised which only uses the input samples while cross entropy uses the output samples as well. This same heuristic is also used for the experiments on real data sets. Further in order to appreciate the method, the non-parametric student based neighborhood was not used, although theoretically one could use any useful neighborhood function.

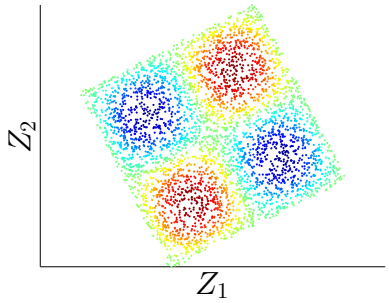
4.4.1 Regression: Parity data

The best value of continuity from cross-validations indicate that their performance is similar to standard SDPP and the other variants introduced in the previous chap-

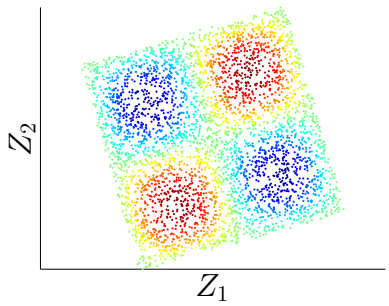
ter. As a note, the continuity measures are basically designed with distances and not probabilities, however they may still be relatively indicative. Figures 4.2(b), 4.2(c) and 4.2(d) show the visualizations produced by the SPPP Gaussian, Heavy-tail and Linear respectively. It can be easily seen that the methods have identified the correct two principal variables that contribute to the responses. Of the three methods the Linear is capable of retaining the same amount of skewness as in the original data. Interestingly when the student distribution was used as a mere neighborhood selection method (as in the previous chapter), it did not produce a good visualization. This implies that probabilities produce better results when used alone rather than in combination with pairwise distances, because the probabilities in this case are already normalized by their density in the projection space during optimization unlike the case in SDPP where the neighborhood matrix is fixed once it is computed.



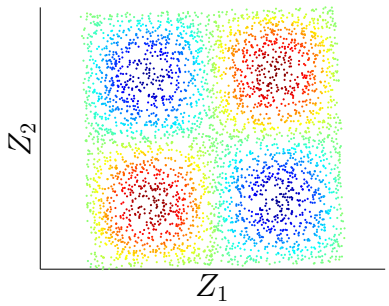
(a) **CV**



(b) **SPPP-Gaussian**



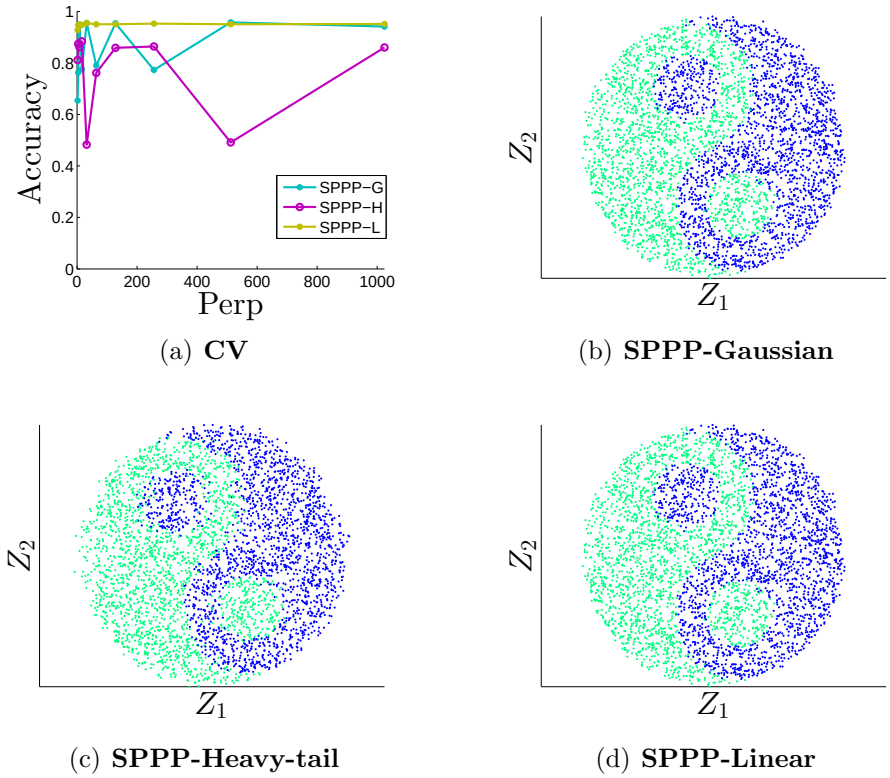
(c) **SPPP-Heavy-tail**



(d) **SPPP-Linear**

Figure 4.2: SPPP for regression

4.4.2 Classification: Taijitu Symbol



(a) **CV** (b) **SPPP-Gaussian**

(c) **SPPP-Heavy-tail** (d) **SPPP-Linear**

Figure 4.3: SPPP for classification

The cross-validation chart is shown in figure 4.3(a) where one can see that the best corresponding parameters of the three methods obtain almost similar accuracies. The visually pleasing projections are shown in figures 4.3(b), 4.3(c) and 4.3(d). Especially SPPP-Gaussian (SPPP-G) and SPPP-Linear (SPPP-L) seem to retain the same amount of skewness and rotation as in the original data which is essential to note in order to trust them when applying to data sets where the true intrinsic dimensionality or the local geometry is not known beforehand.

Chapter 5

Experimental Verifications

In this chapter we provide experimental verifications of all the contributions of this thesis namely 3 neighborhood functions for SDPP and SPPP (3 versions) itself while comparing their performance against a few of the recent state-of-the-art techniques such as PLS, KPLS, KDR, SDPP (supervised DR), PCA and KPCA (unsupervised DR). The interested reader is referred to the appendix for an overview of the compared supervised DR methods. First regression followed by classification tasks are carried out on a few well known UCI repository data sets [2]. To facilitate accommodating projections, for all the experiments 2 dimensions is fixed to be the desired reduced space.

5.1 Experimental setup

Each of the techniques we use have their own special parameter that needs to be found using cross-validation on each data set separately. For this reason we perform a 5-fold cross-validation over a range of 10 candidate parameters for every technique. The kernel methods (KDR,KPLS and KPCA) use RBF Gaussian kernel whose spread (*Kernel width*) is found using cross-validation. A common heuristic to set the spread of RBF kernels is along the median of the data set. For this reason our candidates range has a maximum of twice the median reached in 10 equally-spaced steps. KDR requires kernel widths on both input and the response space which makes 10^2 combinations to be evaluated. The standard version of SDPP requires the number of neighbors while the entropy, cross entropy and all three versions of SPPP require their corresponding perplexity values. The candidate parameters for SDPP and our perplexity-parameterized methods follow naturally from the previous chapters. The choice of computing the neighborhood matrix for SPPP versions, which is to use entropy or cross entropy was made as

SDPP-Entropy	SDPP-Cross Entropy	SDPP-Student	CV for perplexity
SPPP-Gaussian	SPPP-Heavy-tail	SPPP-Linear	CV for perplexity
KDR	KPLS	KPCA	CV for kernel width
SDPP-standard	PLS	PCA	CV for neighbors

Table 5.1: Image matrix

explained in the previous chapter. The student distribution was not made use of due to its non-parametric nature. Again the cross-validation in SPPP only corresponds to the perplexity estimation within $\mathbf{G}$ since the variances in $\mathbf{P}$ and $\mathbf{Q}$ (for Gaussian) were set to a constant as explained earlier. The compared non parametric methods are SDPP-student, PCA and PLS.

Since we do not explicitly have a test set, we learn the transformation matrix using the training partition and evaluate the performance on the projection obtained on the test partition for each fold. For regression the measure is continuity from the $\mathcal{Z}$ to $\mathcal{Y}$ space [1] and for classification it is 1-nearest neighbor (or simply nn) accuracy. As a preprocessing step, all data sets were mean centered. The procedure and presentation of results follow the same steps established in chapter 2, this time cumulative of all five folds.

The results are organized in a manner that allows showing the cross-validation performance alongside their corresponding 2D projections. The figure placements includes an image matrix format as shown in table 5.1. The first row contains the probabilistic neighborhoods of SDPP including the non-parametric student based neighborhood. The second row displays the fully probabilistic methods, as in SPPP, followed by the third row of kernel methods. The last row shows SDPP-standard and two other non parametric methods PLS and PCA. For the reader's convenience all measures on the y axis have been set between 0 and 1 in all experiments.

5.2 Regression on UCI data

Table 5.2[2] shows the description of the data sets used for regression. We follow the experimental setup and show the results for cross-validation and the projections for each data set separately. Finally we comment on their performances for regres-

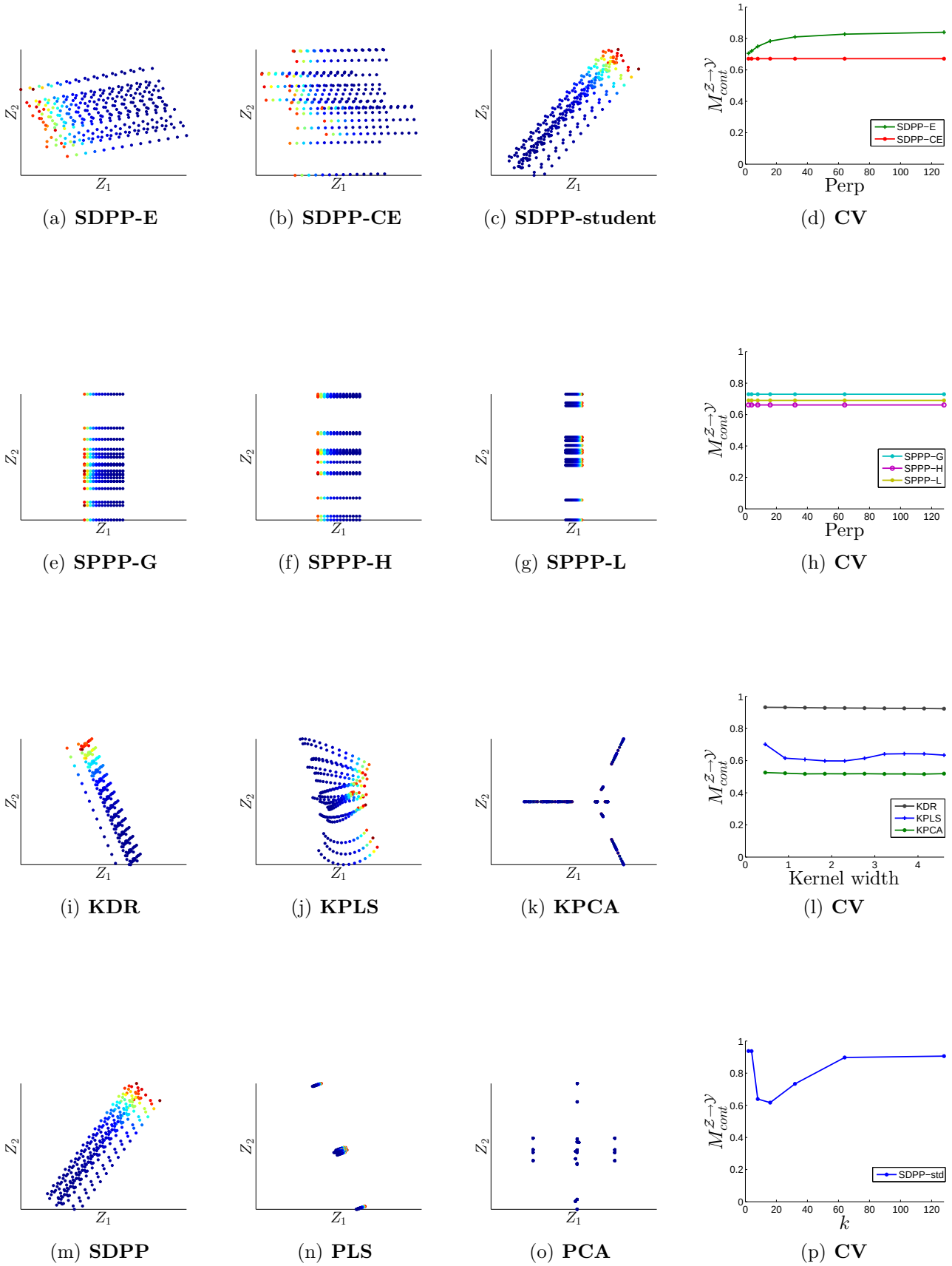
[1] Continuity measure are still based on pairwise distances and hence should not be taken too seriously for the probabilistic methods, although, they can be fairly indicative.

[2] n- Sample size and d- Initial dimensions

Data set	**n**	**d**
Yacht Hydrodynamics	308	6
Housing	506	13
Concrete compressive strength	1030	8
Airfoil self-noise	1503	5

Table 5.2: Regression data sets

sion. For Concrete compressive strength and Airfoil self-noise data sets KDR could not be run due to high computational loads involving high dimensional matrices.



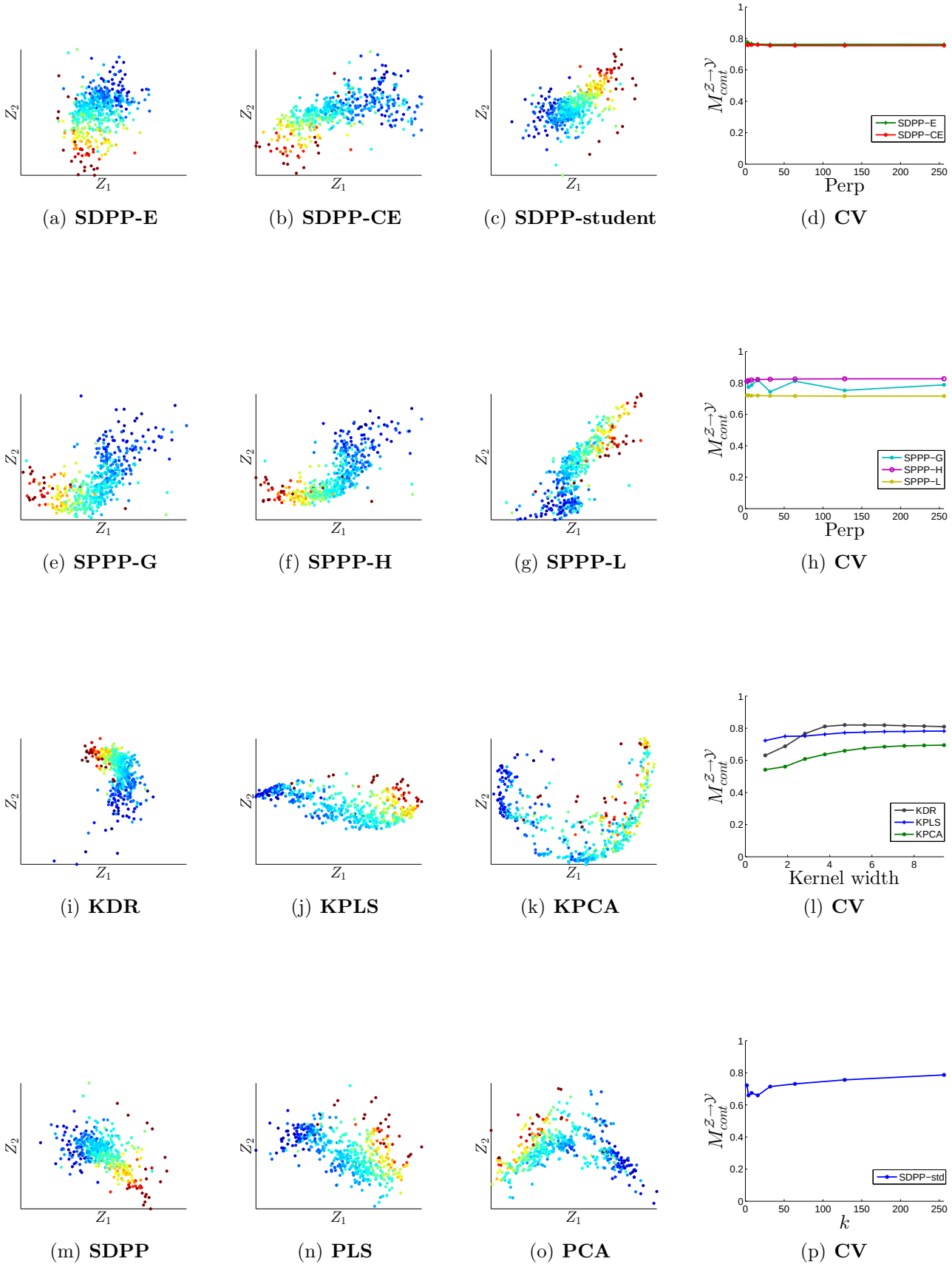
(a) **SDPP-E** (b) **SDPP-CE** (c) **SDPP-student** (d) **CV**

(e) **SPPP-G** (f) **SPPP-H** (g) **SPPP-L** (h) **CV**

(i) **KDR** (j) **KPLS** (k) **KPCA** (l) **CV**

(m) **SDPP** (n) **PLS** (o) **PCA** (p) **CV**

Figure 5.1: Yacht Hydrodynamics



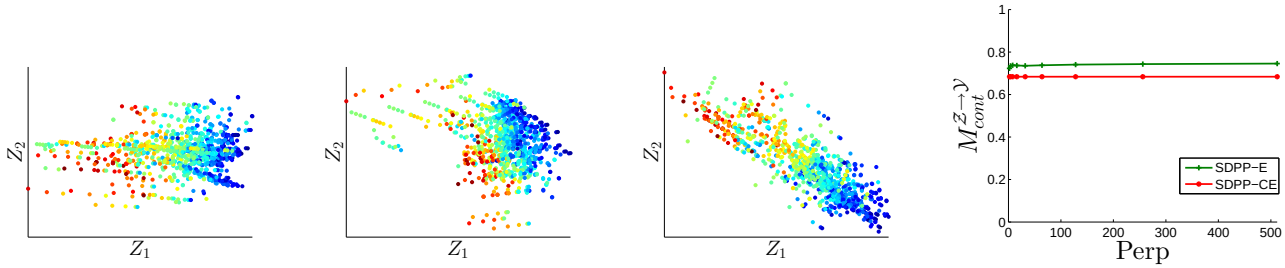
(a) **SDPP-E** (b) **SDPP-CE** (c) **SDPP-student** (d) **CV**

(e) **SPPP-G** (f) **SPPP-H** (g) **SPPP-L** (h) **CV**

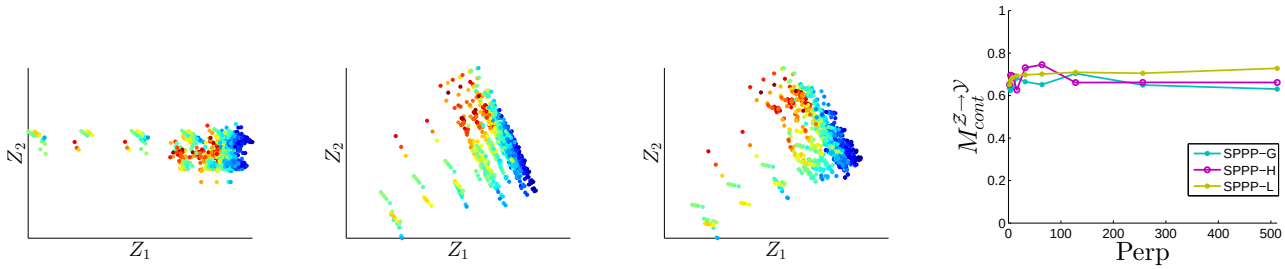
(i) **KDR** (j) **KPLS** (k) **KPCA** (l) **CV**

(m) **SDPP** (n) **PLS** (o) **PCA** (p) **CV**

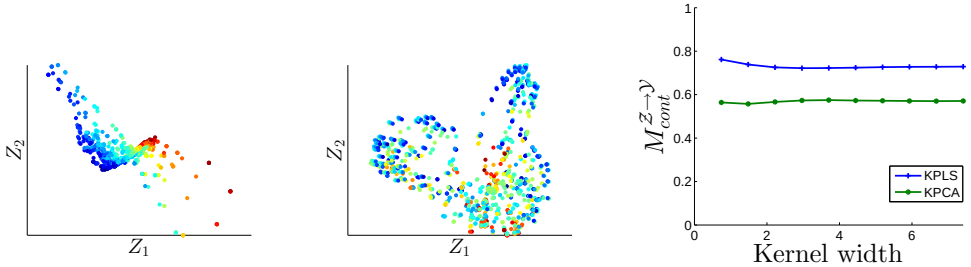
Figure 5.2: Housing



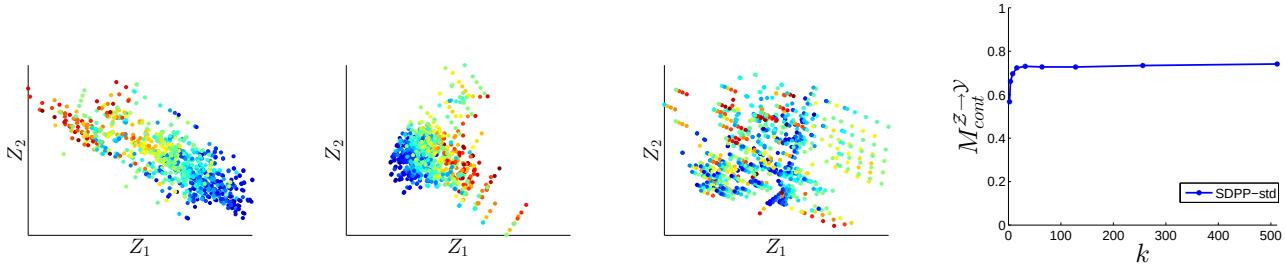
(a) **SDPP-E** (b) **SDPP-CE** (c) **SDPP-student** (d) **CV**



(e) **SPPP-G** (f) **SPPP-H** (g) **SPPP-L** (h) **CV**

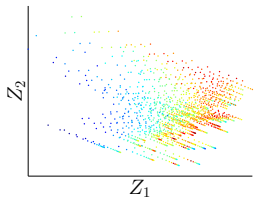


(i) **KPLS** (j) **KPCA** (k) **CV**

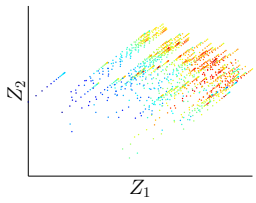


(l) **SDPP** (m) **PLS** (n) **PCA** (o) **CV**

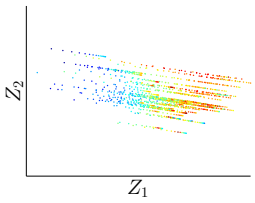
Figure 5.3: Concrete compressive strength



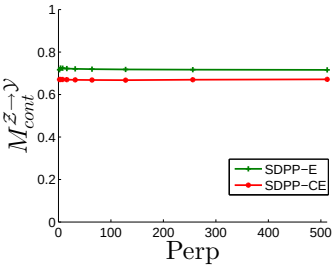
(a) **SDPP-E**



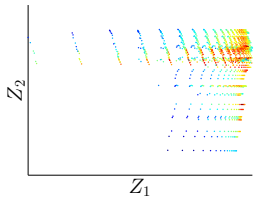
(b) **SDPP-CE**



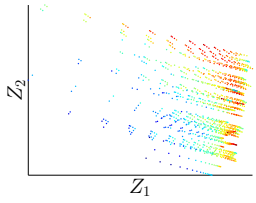
(c) **SDPP-student**



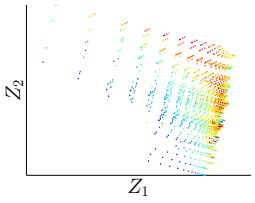
(d) **CV**



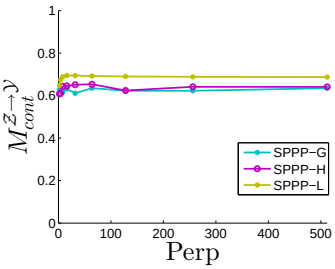
(e) **SPPP-G**



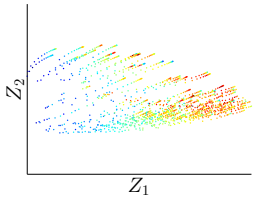
(f) **SPPP-H**



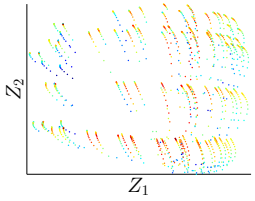
(g) **SPPP-L**



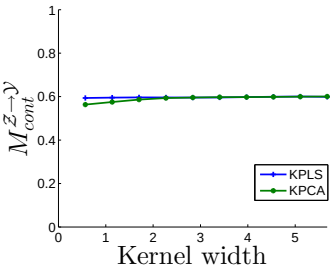
(h) **CV**



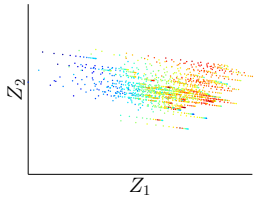
(i) **KPLS**



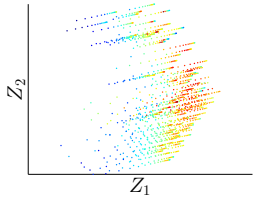
(j) **KPCA**



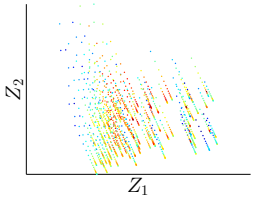
(k) **CV**



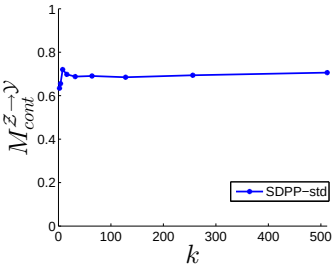
(l) **SDPP**



(m) **PLS**



(n) **PCA**



(o) **CV**

Figure 5.4: Airfoil self-noise

5.2.1 Discussions

For this discussion we have four data sets in a regression setting. On a general level we can notice that almost all methods perform well given their own limitations in design. However it is interesting to note the competitive nature of the entropy and cross entropy based neighborhood functions of SDPP (precisely SDPP-E and SDPP-CE). From the cross-validation charts one can easily see the relatively consistent and smooth performance over a range of perplexity values unlike the fluctuations as in the standard version of SDPP. This is indicative of the smooth neighborhood sizes that penalizes the cost function gradually thereby allowing a certain hit-or-miss range in the neighborhood selection. Both the weighting functions also produce similar although not the same visualizations. This indicates that the datasets obey the surjective definitions. Further the corresponding visualizations by the student weighted function closely resembles that of the standard SDPP which is obvious because it has only information about the pairwise distances in its definition. But the advantage however is that it is non-parametric which is useful in cases where cross-validation is an expensive option.

The performance of the best value of perplexity found using cross-validation for the three versions of SPPP show that they are equivalent or better in cases compared to other methods. Each one of them have unique visualizations by virtue of their chosen distribution. Although the continuity measure for probabilistic methods do not best quantify their performance, the visualizations indicate that they are worth building regression models with. Additionally SPPP methods are relatively smooth performing, following their probabilistic tendencies. This is quite useful for sparse data sets indicating their relative robustness towards noise.

Now let us compare the results to the three kernel methods, KDR and KPLS (supervised) and KPCA (unsupervised). The kernel methods are generally known for their ability to allow nonlinear mapping in the projection space, but SDPP variants and SPPP versions basically only compute a linear mapping. Despite this limitation, the results seem to indicate promising competition, at least for the data sets considered here. Of the 5 perplexity-parameterized methods introduced in this thesis, it has to be said that SDPP-E and SDPP-CE are at times consistently better performing than the three versions of SPPP. Their 2D projections provide good basis for visual inspection and analysis. Amongst the methods chosen for comparison, KPLS presents interesting visualizations.

In addition to producing pleasing visualizations and retaining good continuity values, SPPP also preserves internal similarities faithfully that would allow us to further build a model. Next we show the results of SPPP on classification tasks.

Data set	**n**	**d**	**Classes**
Wine	178	13	3
Glass Identification	214	10	6
Balance scale	625	4	3
Waveform Database Generator (Version 1)	5000	21	3

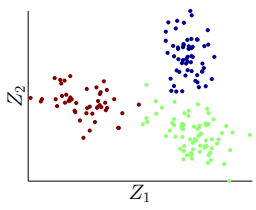
Table 5.3: Classification data sets

5.3 Classification on UCI data

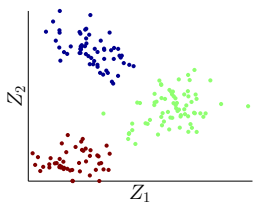
While retaining the same formulation, SPPP can also be used for classification tasks with a slight modification in encoding the class information as discussed earlier. This is because in a regression setting, the responses are true measures, while in the classification setting they are arbitrary values indicating class labels. The data sets used for classification are described in the table 5.3 [3].

Following the experimental procedure we show the results of cross-validation with good classification accuracy as the target and their corresponding 2D projections. Except for the Waveform data set we were able to run KDR on all others.

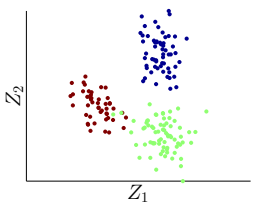
[3] n- Sample size and d- Initial dimensions



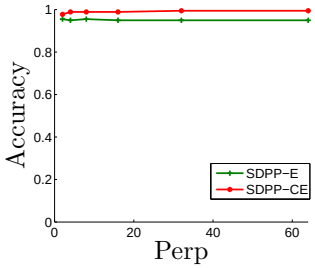
(a) **SDPP-E**



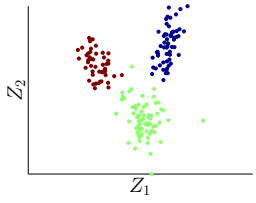
(b) **SDPP-CE**



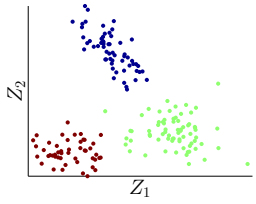
(c) **SDPP-student**



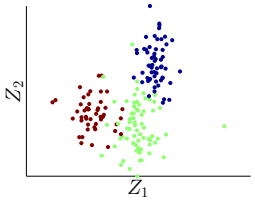
(d) **CV**



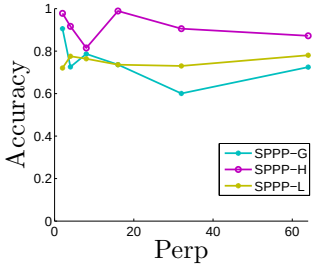
(e) **SPPP-G**



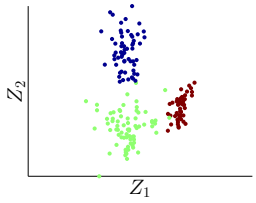
(f) **SPPP-H**



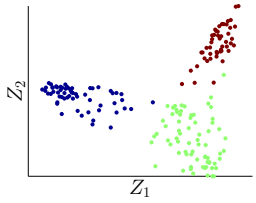
(g) **SPPP-L**



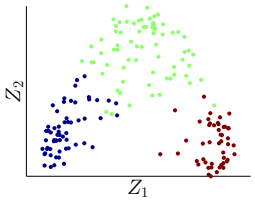
(h) **CV**



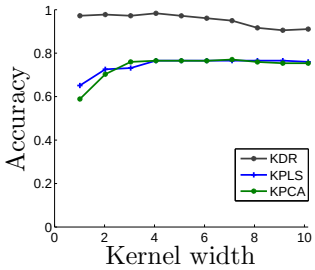
(i) **KDR**



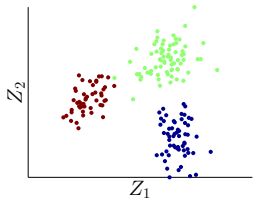
(j) **KPLS**



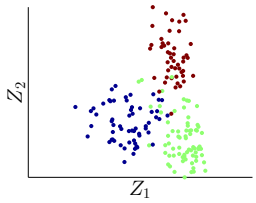
(k) **KPCA**



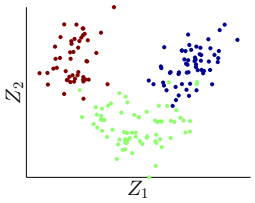
(l) **CV**



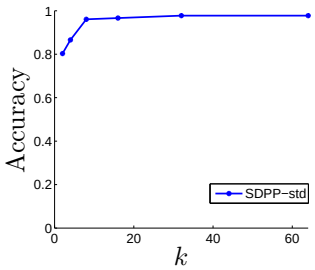
(m) **SDPP**



(n) **PLS**

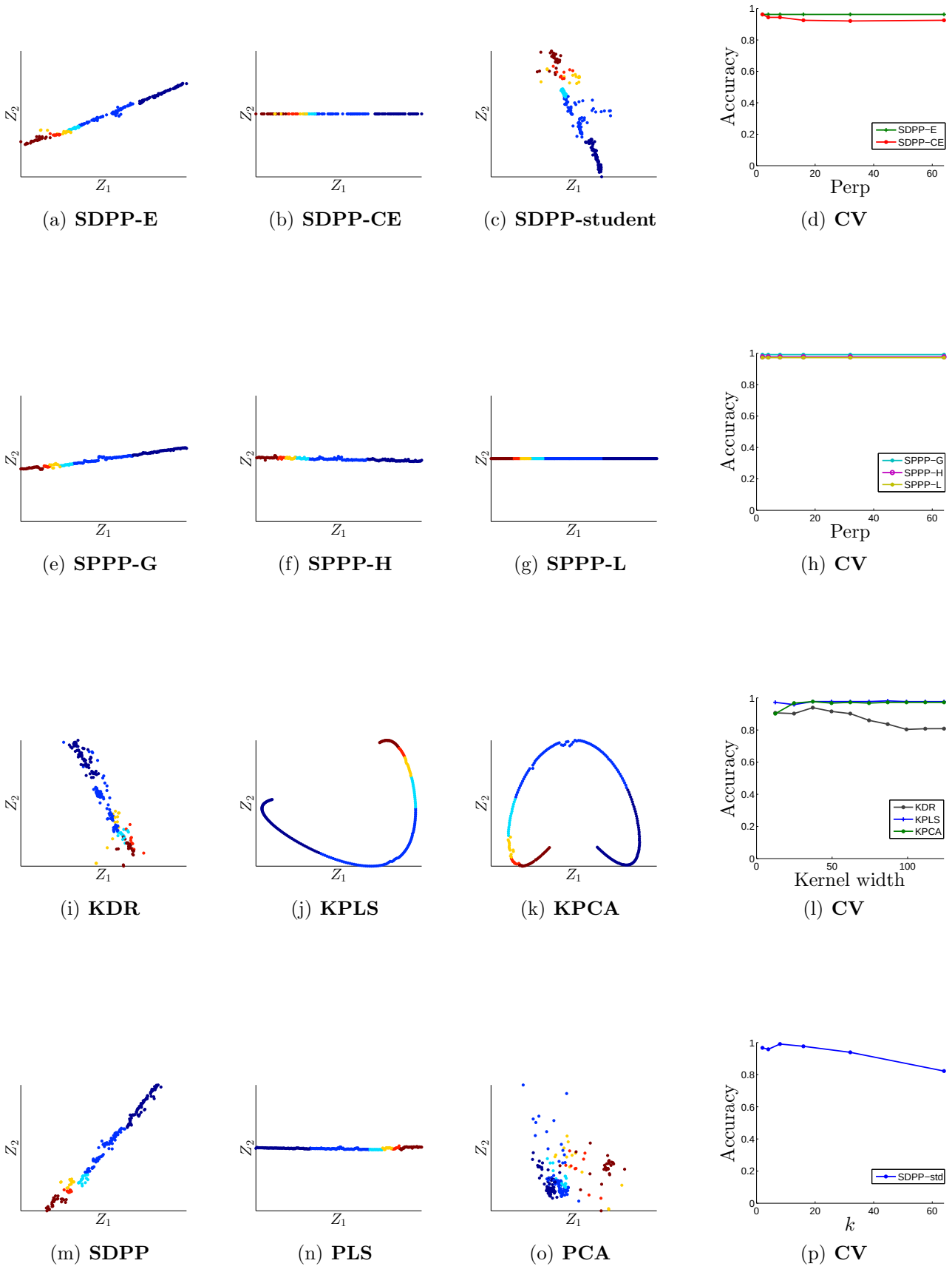


(o) **PCA**



(p) **CV**

Figure 5.5: Wine



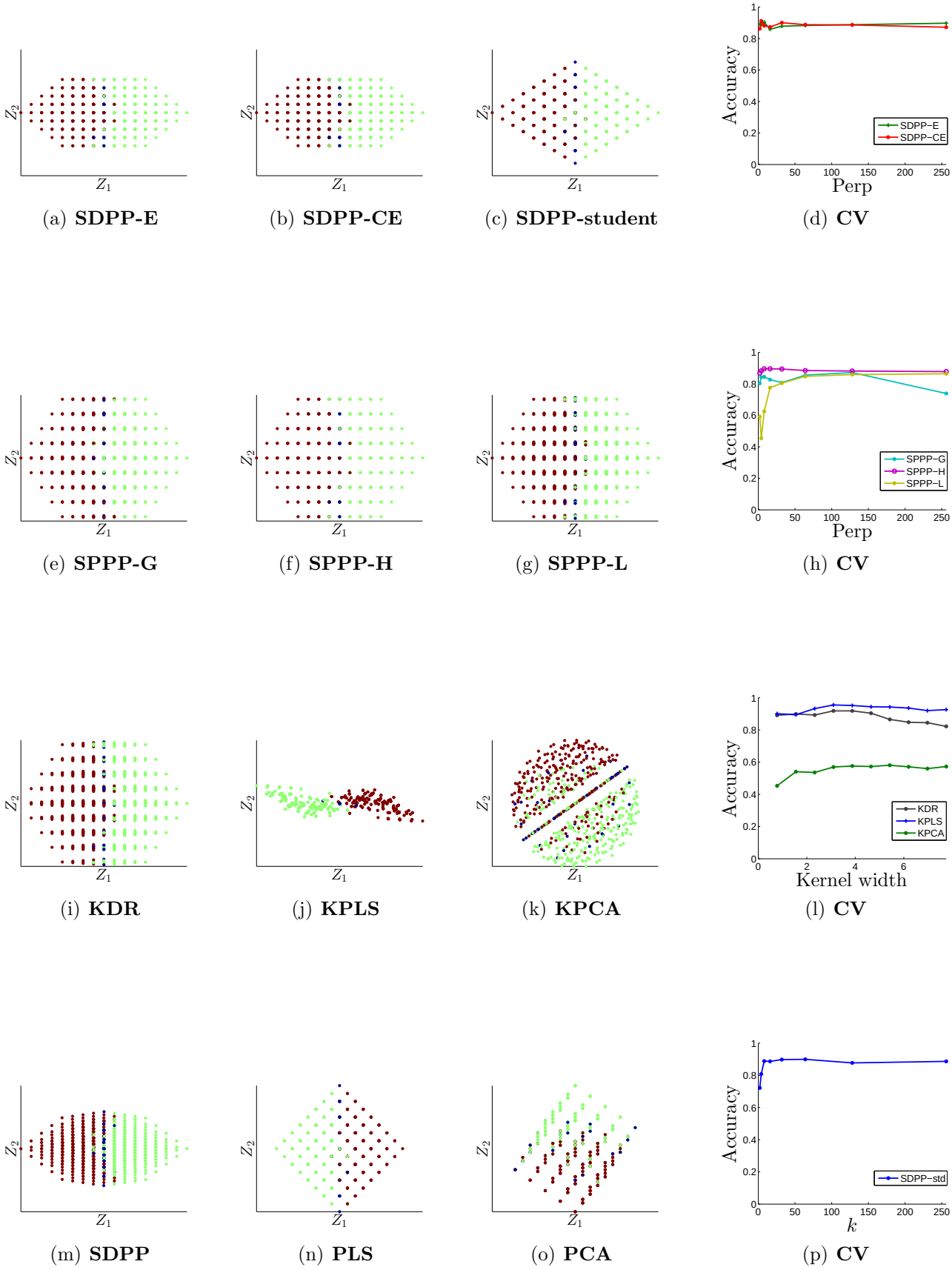
(a) **SDPP-E** (b) **SDPP-CE** (c) **SDPP-student** (d) **CV**

(e) **SPPP-G** (f) **SPPP-H** (g) **SPPP-L** (h) **CV**

(i) **KDR** (j) **KPLS** (k) **KPCA** (l) **CV**

(m) **SDPP** (n) **PLS** (o) **PCA** (p) **CV**

Figure 5.6: Glass Identification



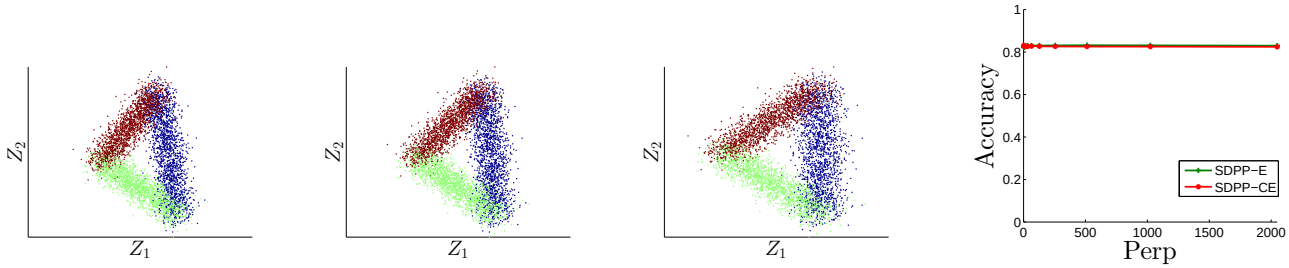
(a) **SDPP-E** (b) **SDPP-CE** (c) **SDPP-student** (d) **CV**

(e) **SPPP-G** (f) **SPPP-H** (g) **SPPP-L** (h) **CV**

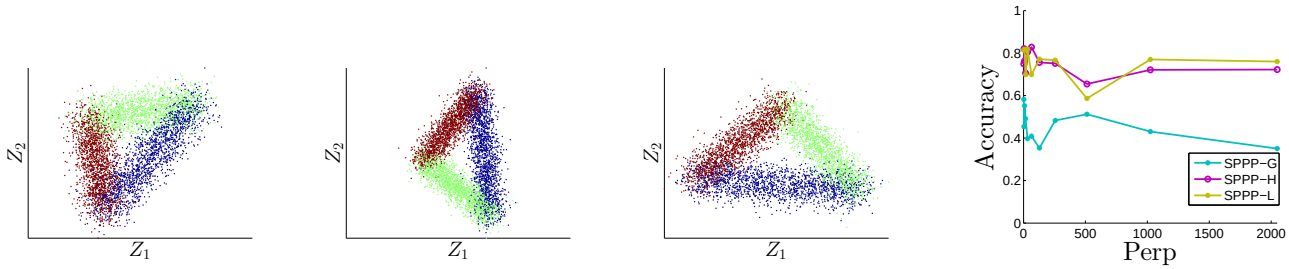
(i) **KDR** (j) **KPLS** (k) **KPCA** (l) **CV**

(m) **SDPP** (n) **PLS** (o) **PCA** (p) **CV**

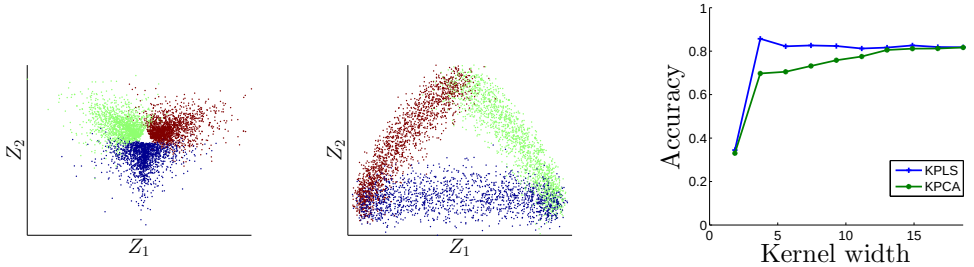
Figure 5.7: Balance scale



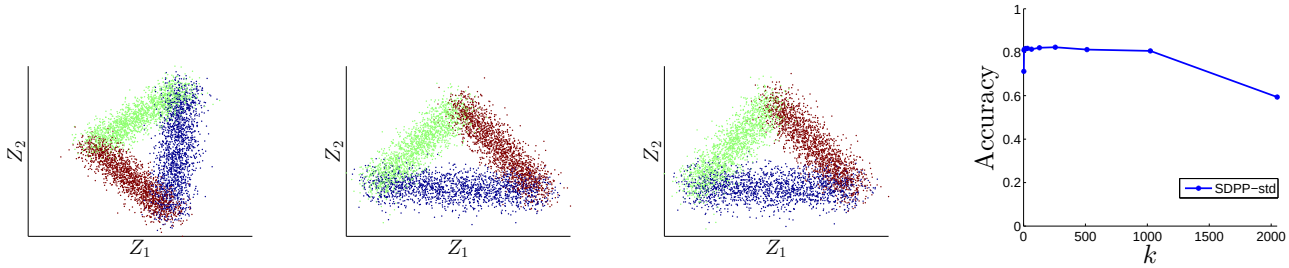
(a) **SDPP-E** (b) **SDPP-CE** (c) **SDPP-student** (d) **CV**



(e) **SPPP-G** (f) **SPPP-H** (g) **SPPP-L** (h) **CV**



(i) **KPLS** (j) **KPCA** (k) **CV**



(l) **SDPP** (m) **PLS** (n) **PCA** (o) **CV**

Figure 5.8: Waveform Database Generator (Version 1)

5.3.1 Discussions

For this discussion we have classification tasks performed on four varied data sets. From the cross-validation charts we can again see the relatively smooth performance of the SDPP-E and SDPP-CE unlike the fluctuations as in the standard SDPP. The reason remains the same thereby showing evidences of relaxed penalizations in using probabilities instead of binary valued neighborhoods. The visualizations also indicate that all three weighting functions are better in some cases than standard SDPP for the data sets considered.

The classification performance of the SPPP versions especially the Heavy-tail and Linear, display highly competitive performance compared to all other methods. A measure so strict as $1nn$ accuracy would require maintaining the internal structure so precisely since a slight perturbation in the arrangement could affect the performance. Despite this SPPP have managed to reach good measures. One can also observe that all the three methods consistently produce pleasing visualizations if not equivalent, especially SPPP-L for Glass identification data set.

Despite the Waveform data set having a large sample size, SDPP-E and SDPP-CE have shown remarkable consistency through a range of parameter choices even topping the kernel methods. It is difficult to state at least from this comparison as to which one is better of the two. Additionally for all the experiments considered here, SDPP-student has produced similar 2D projections as the standard SDPP, but with the advantage of being completely non-parametric. Once again amongst the compared methods, KPLS visualizations are interesting.

Although all the three versions of SPPP have performed quite well on most data sets in both regression and classification tasks, the Heavy-tail function is more promising. Firstly because it has consistently produced competitive performance for the data sets considered here. Secondly as also pointed out in [32] it allows obtaining nice visualizations. This is understandable because the Heavy-tail is a steep function with a long tail thereby allowing to assign more weightage to closeby points and smaller values to the wider ones. Unlike SPPP-H the SPPP-L is influenced more by the pairwise distances instead of assuming a specific distribution to follow. This makes it a classic combination of both distance based and probability based methods.

Chapter 6

Conclusions

The aim of this thesis was to develop a supervised DR technique that maintains relations between points in a stochastic manner. Precisely SPPP learns a linear transformation matrix leading to a projection space where the differences between the probabilistic similarities of the input covariates and their responses are minimized.

We began by suggesting three probabilistic neighborhood functions based on entropy, cross entropy and student's t-distribution as neighborhood selection strategies for a recently proposed method called SDPP. The results on synthetic experiments led to the motivation of formulating a stand alone probabilistic method called SPPP. The three flavors of SPPP introduced in this thesis are namely Gaussian, Heavy-tail and Linear. The common motivation and formulation were presented in addition to their separate gradients. We showed using experiments on both synthetic and real world data sets that SPPP is competitive in its performance for both regression and classification tasks alike. The comparisons included recent state-of-the-art methods such as PLS, KPLS, KDR and SDPP itself (supervised DR) and PCA, KPCA (unsupervised DR).

The benefits of using SPPP are that it maintains probabilistic relations with a smooth number of neighbors unlike that of SDPP which is limited by discrete binary values. This notion gives the edge to SPPP in handling sparse data sets. For regression tasks, SPPP methods often provide good reduced representations as we observed from their visualizations. Consequently one could build simple regression models out of these representations. Similarly for classification as well they show promising separations of classes even to human eyes.

Amongst the contributions of this thesis, the SDPP-E and SDPP-CE are quite competitive, sometimes even better than SPPP. Moreover we noticed that of the

three methods of SPPP the Linear is a classic combination of both distance based and probability based methods since it does not assume a distribution in its own sense and follows the data. Of the three versions, however SPPP-H provides both nice visualizations and performance.

Possible improvements and research directions:

- While using the simple weighted individual differences model proves worthy, it would be interesting to see how the divergence measures assist in the process such as KL, Itakura-Saito divergence or other Bregman divergences.
- We used only Euclidean distances in our method, but this does not include the manifold information. One could consider using graphs constructed along manifolds. This might give an appearance of a 'kernel' version of the same method.
- We used entropy and cross entropy in our experiments. But there may be cases where some data set might violate the surjective definition. It would be interesting to understand and device methods to overcome this difficulty or experiment with the use of conditional entropy instead of cross entropy.

Chapter 7

Appendix

7.1 A brief overview of PLS, KPLS and KDR

Partial Least Squares (PLS)

Partial Least Squares is a method for finding the linear relationship between a set of input covariates and their responses [38]. PLS uses a set of linear equations,

$$\mathbf{X} = \mathbf{Z}\mathbf{A}^T + \mathbf{E}$$
$$\mathbf{Y} = \mathbf{T}\mathbf{B}^T + \mathbf{F}$$

where $\mathbf{X}$ and $\mathbf{Y}$ are input covariates and their responses respectively and $\mathbf{A}$ and $\mathbf{B}$ are loading matrices. $\mathbf{Z}_{n*r}$ and $\mathbf{T}_{n*r}$ are the score matrices while $\mathbf{E}$ and $\mathbf{F}$ are residuals. The objective of PLS is to find projection matrices $\mathbf{W} = [\mathbf{w}_1, ...\mathbf{w}_r]$ and $\mathbf{U} = [\mathbf{u}_1, ...\mathbf{u}_r]$ that maximizes the covariances between the projected inputs and the responses.

$$\max_{\mathbf{w}_i, \mathbf{u}_i} [Cov(\mathbf{z}_i, \mathbf{t}_i)]^2 = \max_{\mathbf{w}_i, \mathbf{u}_i} [Cov(\mathbf{X}\mathbf{w}_i, \mathbf{Y}\mathbf{u}_i)]^2 \tag{7.1}$$

where the $Cov(\mathbf{z}_i, \mathbf{t}_i) = \mathbf{z}_i^T \mathbf{t}_i / n$ denotes the sample covariance between the score vectors $\mathbf{z}$ and $\mathbf{t}$. To this end, PLS uses an iterative approach that deflates $\mathbf{X}$ and $\mathbf{Y}$ after extracting $\mathbf{z}$ and $\mathbf{t}$ until no more deflating is possible or convergence reached. The NIPALS algorithm [38] is an example of this approach.

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. $\mathbf{t} = \mathbf{Y}\mathbf{u}$

5. $\mathbf{X} \leftarrow \mathbf{X} - \mathbf{z}\mathbf{z}^T\mathbf{X}$ and $\mathbf{Y} \leftarrow \mathbf{Y} - \mathbf{t}\mathbf{t}^T\mathbf{Y}$

Consequently another approach allows to obtain the latent variables using a single eigenvalue decomposition.

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

$\mathbf{Z}$ and $\mathbf{T}$ are computed as, $\mathbf{Z} = \mathbf{X}\mathbf{W}$ and $\mathbf{T} = \mathbf{Y}\mathbf{U}$. Interestingly both the above approaches lead to the same solution [14].

Kernel PLS (KPLS)

The limitation of PLS in handling nonlinear data is overcome by the kernel version called Kernel PLS (KPLS) [22]. The nonlinear data is mapped to a high dimensional space $\mathcal{F}$ corresponding to a reproducing kernel Hilbert Space such that $\mathbf{x} \rightarrow \phi(\mathbf{x})$.

The *kernel trick* reduces the problem once again to a set of linear algebraic equations as in PLS. Thanks to the fact that a value of the dot product between two vectors in $\mathcal{F}$ allows them to be evaluated by the kernel function.

$$k(\mathbf{x}, \mathbf{y}) = \boldsymbol{\phi}\mathbf{x}^T\boldsymbol{\phi}\mathbf{y}, \quad \forall\ \mathbf{x}, \mathbf{y} \in X$$

The gram matrix $\mathbf{K}$ is defined as the cross dot product of all the mapped input data points, $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \langle\boldsymbol{\phi}(\mathbf{x}_i, \mathbf{x}_j)\rangle \implies \mathbf{K} = \boldsymbol{\phi}^T\boldsymbol{\phi}$ where $\boldsymbol{\phi} = [\boldsymbol{\phi}(\mathbf{x}_1).....\boldsymbol{\phi}(\mathbf{x}_n)]$. The modified algorithm now deflates $\mathbf{K}$ and $\mathbf{Y}$ unlike $\mathbf{X}$ and $\mathbf{Y}$ in PLS. This is performed in steps as shown below.

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 steps 2-3 until convergence.
5. Deflate $\mathbf{K} \leftarrow (\boldsymbol{\phi} - \mathbf{z}\mathbf{z}^T\boldsymbol{\phi})^T(\boldsymbol{\phi} - \mathbf{z}\mathbf{z}^T\boldsymbol{\phi})$ and $\mathbf{Y} \leftarrow \mathbf{Y} - \mathbf{t}\mathbf{t}^T\mathbf{Y}$

This is performed until the rank of $\mathbf{K}$ is reached as mentioned in the modified NIPALS algorithm [22].

Kernel Dimensionality Reduction (KDR)

KDR [8] is a supervised DR approach for sufficient dimensionality reduction (SDR). KDR tries to find an orthogonal transformation $\mathbf{W}$ such that $\mathbf{Y} \perp\!\!\!\perp \mathbf{X} \mid \mathbf{W}^T\mathbf{X}$ [1]

The different SDR algorithms are aimed at solving the inverse regression problem which is to find the expectation $\mathbf{E}(\mathbf{X} \mid \mathbf{Y})$ based on the fact that if the conditional distribution $\mathbf{P}(\mathbf{Y} \mid \mathbf{X})$ varies along a subspace $\mathcal{X}$ then the inverse regression $\mathbf{E}(\mathbf{X} \mid \mathbf{Y})$ also should lie along $\mathcal{X}$. Examples of SDR include Sliced Invariance Regression (SIR), principal Hessian Direction (pHd), sliced average variance estimation (SAVE) and contour regression. However all these algorithms are limited by their assumption about the marginal distribution $\mathbf{P}_\mathbf{X}(\mathbf{x})$, such as the distribution is elliptical. This imposes serious limitations when not being the case.

KDR was proposed as a method that overcomes such assumptions and instead converts the problem of imposing conditional independence to minimizing the conditional covariance operator in the Reproducing Kernel Hilbert Space (RKHS), $\mathbf{H}_\mathbf{X}$ and $\mathbf{H}_\mathbf{Y}$. By assigning a Lebesque measure of probability spaces over which $\mathbf{X}$ and $\mathbf{Y}$ are defined (such that $f_1, f_2 \in \mathbf{H}_\mathbf{X}, \langle f_1, f_2 \rangle = \int f_1(\mathbf{x}) f_2(\mathbf{x}) d\mathbf{P}(\mathbf{X})$), a cross covariance operator can be defined as,

$$\langle g, \Sigma_{\mathbf{YX}} f \rangle = \mathbf{E}_{\mathbf{XY}}[(f(\mathbf{X}) - \mathbf{E}_\mathbf{X}[f(\mathbf{X})])(g(\mathbf{Y}) - \mathbf{E}_\mathbf{Y}[g(\mathbf{Y})])]$$

A conditional covariance operator is then defined as $\Sigma_{\mathbf{YY} \mid \mathbf{X}} \equiv \Sigma_{\mathbf{YY}} - \Sigma_{\mathbf{YX}}\Sigma_{\mathbf{XX}}^{-1}\Sigma_{\mathbf{XY}}$. Let us introduce an orthonormal transformation operator $\mathbf{W}\mathbf{W}^T$ that spans the subspace corresponding to the input covariates. The kernels of RKHS's $\mathbf{H}_\mathbf{X}$ and $\mathbf{H}_\mathbf{Y}$ are denoted by $\mathbf{K}_\mathbf{X}$ and $\mathbf{K}_\mathbf{Y}$ respectively (such that $\mathbf{K}^\mathbf{W}(\mathbf{x}_1, \mathbf{x}_2) \equiv \mathbf{K}(\mathbf{W}^T\mathbf{x}_1, \mathbf{W}^T\mathbf{x}_2)$). In this setting subject to some weak condition on $\mathbf{H}_\mathbf{X}$, $\mathbf{H}_\mathbf{Y}$ and the probability measures, it can be shown that $\Sigma^\mathbf{W}_{\mathbf{YY} \mid \mathbf{X}} \geq \Sigma_{\mathbf{YY} \mid \mathbf{X}} \Leftrightarrow \mathbf{Y} \perp\!\!\!\perp \mathbf{X} \mid \mathbf{W}^T\mathbf{X}$. [2] [8]. The objective function of KDR thus simplifies to finding $\mathbf{W}$ as the component that minimizes the trace as,

$$Tr[\mathbf{G}_\mathbf{Y}(\mathbf{G}^\mathbf{W}_\mathbf{X} + \mathfrak{m}\epsilon_\mathfrak{m} I_\mathfrak{m})^{-1}]$$

$$\text{s.t} \quad \mathbf{W}^T\mathbf{W} = 1$$

where $\mathbf{G}_\mathbf{Y}$ and $\mathbf{G}^\mathbf{W}_\mathbf{X}$ are the centered gram matrices, $\mathfrak{m}$ is the sample size and ϵ is a regularization parameter. $\mathbf{W}$ is found using gradient descent. Unfortunately the computational load of KDR is high because of the need to compute the inverse of the kernel matrix at each iterative step.

[1] We use an abuse notation of $\mathbf{W}^T\mathbf{X}$ as the projected subspace.

[2] inequality arises due to ordering that can be defined for self-adjoint operators

Bibliography

[1] Amaldi, E., and Kann, V. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science , Vol. 209, No. 1* (1998), 237–260.

[2] Bache, K., and Lichman, M. UCI machine learning repository. *URL http://archive. ics. uci. edu/ml 19* (2013).

[3] Barshan, E., et al. Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds. *Pattern Recognition , Vol. 44, No. 7* (2011), 1357–1371.

[4] Belkin, M., and Niyogi., P. Laplacian eigenmaps and spectral techniques for embedding and clustering. *NIPS , Vol. 14,* (2001), 585–591.

[5] Brito, M. R., et al. Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. *Statistics and Probability Letters , Vol. 35, No. 1* (1997), 33–42.

[6] Cover, T., and Hart., P. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on Vol. 13, No. 1* (1967), 21–27.

[7] Fan, J., and Lv, J. A selective overview of variable selection in high dimensional feature space. *Statistica Sinica Vol. 20, No. 1* (2010), 101–148.

[8] Fukumizu, K., Bach, F., R., and Jordan, M., I. Kernel dimension reduction in regression. *The Annals of Statistics , Vol. 37, No. 4* (2009), 1871–1905.

[9] Gupta, M. R., Garcia, E. K., and Chin, E. Adaptive local linear regression with application to printer color management. *Image Processing, IEEE Transactions , Vol. 17, No. 6* (2008), 936–945.

[10] Guyon, I., and Elisseeff., A. An introduction to variable and feature selection. *The Journal of Machine Learning Research , Vol. 3* (2003), 1157–1182.

[11] Hanley, J. A., and McNeil., B. J. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology , Vol. 143, No. 1* (1982), 29–36.

[12] Hinton, G., E., and Roweis, S., T. Stochastic neighbor embedding. *In Advances in neural information processing systems , Vol. 2* (2002), 833–840.

[13] Hinton, G., E., and Salakhutdinov, R., R. Reducing the dimensionality of data with neural networks. *Science , Vol. 313, No. 5786* (2006), 504–507.

[14] Höskuldsson, A. PLS regression methods. *Journal of chemometrics , Vol. 2, No. 3* (1988), 211–228.

[15] Kohonen, T. The self-organizing map. *Proceedings of the IEEE , Vol. 78, No. 9* (1990), 1464–1480.

[16] Lee, J. A., and Verleysen., M. Nonlinear dimensionality reduction of data manifolds with essential loops. *Neurocomputing Vol. 67* (2005), 29–53.

[17] Lee, J. A., and Verleysen., M. Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomputing , Vol. 72, No. 7* (2009), 1431–1443.

[18] Mokbel, B., et al. Visualizing the quality of dimensionality reduction. *Neurocomputing , Vol. 112* (2013), 109–123.

[19] Pearson, K. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science , Vol. 2, No. 11* (1901), 559–572.

[20] Penrose, M. D. A strong law for the longest edge of the minimal spanning tree. *The Annals of Probability , Vol. 27, No. 1* (1999), 246–260.

[21] Polak, E., and Ribiere., G. Note sur la convergence de méthodes de directions conjuguées.". *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique Et Analyse Numérique , Vol 3, No. R1* (1969), 35–43.

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

[23] Roweis, S. T., and Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science , Vol. 290, No. 5500* (2000), 2323–2326.

[24] Saxena, A., Gupta, A., and Mukerjee., A. Non-linear dimensionality reduction by locally linear isomaps. *Neural Information Processing . Springer Berlin Heidelberg* (2004), 1038–1043.

[25] Schölkopf, B., Smola, A., and Müller, K. R. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation , Vol 10, No. 5* (1998), 1299–1319.

[26] Shannon, C. E. Communication theory of secrecy systems. *Bell system technical journal , Vol 28, No. 4* (1949), 656–715.

[27] Sibson, R. A brief description of natural neighbour interpolation. *Interpreting multivariate data , Vol. 21* (1981).

[28] Takane, Y., Young, F. W., and De Leeuw, J. Nonmetric individual differences multidimensional scaling: an alternating least squares method with optimal scaling features. *Psychometrika , Vol 42, No. 1* (1977), 7–67.

[29] Tenenbaum, J. B., De Silva, V., and Langford., J. C. A global geometric framework for nonlinear dimensionality reduction. *Science , Vol. 290, No. 5500* (2000), 2319–2323.

[30] Torgerson, W. S. Multidimensional scaling: I. Theory and method. *Psychometrika , Vol. 17, No. 4* (1952), 401–419.

[31] Torkkola, K. Feature extraction by non parametric mutual information maximization. *The Journal of Machine Learning Research , Vol 3* (2003), 1415–1438.

[32] van der Maaten, L., and Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research , Vol. 9* (2008), 2579–2605.

[33] van der Maaten, L., Postma, E., O., and van den Herik, H., J. Dimensionality reduction: A comparative review. *Journal of Machine Learning Research , Vol. 10, No. 1-41* (2009), 66–71.

[34] Venna, J., et al. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *The Journal of Machine Learning Research , Vol 11* (2010), 451–490.

[35] Venna, J., and Kaski, S. Local multidimensional scaling. *Neural Networks , Vol 19, No. 6* (2006), 889–899.

[36] Venna, J., and Kaski, S. Comparison of visualization methods for an atlas of gene expression data sets. *Information Visualization , Vol. 6, No. 2* (2007), 139–154.

[37] Weinberger, K. Q., Sha, F., and Saul, L. K. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the twenty-first international conference on Machine learning, ACM* (2004), p. 106.

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

[39] Zhu, Z., Similä, T., and Corona, F. Supervised Distance Preserving Projections. *Neural processing letters , Vol. 38, No. 3* (2013), 445–463.