Mohamed VI Polytechnique University

Modeling, simulation and data analysis

End of Studies Project

A comprehensive overview of kernel principal component analysis from a distance-based viewpoint

Author:
Ismail BACHCHAR

Advisor:
Diego Hernán Peluffo-Ordóñez

Requirement for obtaining the degree of Data Science.

Ben Guerir, Morocco - July 30th, 2021

Dedication

"To my lovely family..."

Acknowledgments

I would like to express my very great appreciation to my advisor Mr. Diego Hernán Peluffo-Ordóñez for his valuable and constructive suggestions during the planning and development of this research work. His willingness to give his time so generously has been very much appreciated. I would also like to thank all people of the following organizations: Department AL-KHWARIZMI at UM6P, SDAS Research Group (`www.sdas-group.com`) and MSDA Research Group (`https://msda.um6p.ma`).

Abstract

The idea of dimensionality reduction (DR) is of transforming the original data from input space to a reduced (low-dim) space. In other words, representing the original data in a new low-dimensional space. This is done by respecting some predefined criteria, such as data-structure preservation or class separation. Such a benefit of data being represented in low-dimensions could be for visualization purposes or even a necessary pre-processing for other downstream data analysis techniques. Several methods are being developed for the DR, the most known ones are the kernel functions. they are very useful because of their quick adaptation to tremendous methods, such as Kernel Support vector machine (SVM), Kernel Principal Component Analysis(K-PCA), and much more.

Keywords: Dimensionality reduction, Kernel functions, Kernel PCA, Feature space.

Contents

Dedication i

Acknowledgments ii

Abstract iii

Contents iv

List of Tables vi

List of Figures vii

1 Introduction 1

2 Background 2
2.1 High-dimensional Data . . . 2
2.1.1 Big Data . . . 2
2.1.2 Today's Challenges . . . 3
2.2 Dimensionality Reduction . . . 4
2.2.1 Class Separability Criterion . . . 4
2.2.2 Structure Preservation . . . 7

3 Overview on Kernels for Dimensionality Reduction 9
3.1 Basics on Dimensionality Reduction . . . 9
3.1.1 Spectral Analysis . . . 9
3.1.2 Principal component analysis (PCA) from a distance-based viewpoint 10
3.2 Kernel Functions . . . 11
3.2.1 The Concept of Kernel . . . 11
3.2.2 Kernel matrix . . . 12
3.2.3 Kernel Trick . . . 12
3.2.4 Types of Kernel . . . 12
3.3 Kernel PCA . . . 13
3.3.1 Feature Space (High-dimensional Space) . . . 13
3.3.2 Kernel PCA from Distance-based Viewpoint . . . 14

4 Experimental Setup **16**

5 Results and Discussion **18**
5.1 Experiment 1 : `Helix` Dataset 18
5.1.1 Structure Preservation Results 19
5.1.2 Class Separability Results 20
5.2 Experiment 2 : `SwissRoll` Dataset 20
5.2.1 Trial 1: 22
5.2.2 Structure Preservation Results 22
5.2.3 Class Separability Results 23
5.2.4 Trial 2: 24
5.2.5 Structure Preservation Results 24
5.2.6 Class Separability Results 25
5.2.7 Trial 3: 26
5.2.8 Structure Preservation Results 26
5.2.9 Class Separability Results 27
5.2.10 Trial 4: 28
5.2.11 Structure Preservation Results 28
5.2.12 Class Separability Results 29
5.3 Experiment 3 : `Toroidal` Dataset 29
5.3.1 Structure Preservation Results 30
5.3.2 Class Separability Results 31

6 Conclusion and Future Work **32**

Bibliography **33**

Appendices **35**

A Demonstration of the inner product equivalence **36**

B Matlab Interface **37**

List of Tables

5.1 Trial settings for `Helix` dataset 19
5.2 Trial settings for `SwissRoll` dataset 21
5.3 Trial settings for `Toroid` dataset 29

List of Figures

2.1 *3 Vs of Big Data.* . . . 3
2.2 *Half Moon original dataset (2D)* . . . 5
2.3 *Half Moon reduced dataset (1D)* . . . 5
2.4 *A curled plane: the Swiss Roll.* . . . 7

3.1 *Non-linear binary classification problem in 2D.* . . . 14
3.2 *Linearly-separable classification problem in higher dimensions (3D).* . . . 14

4.1 *Proposed experimental setup to compare kernel matrices on dimensionality reduction tasks using Kernel PCA.* . . . 16

5.1 Experiment 1: `Helix` Dataset Normalized . . . 18
5.2 Structure Preservation Results for `Helix` Dataset Trial 1 . . . 19
5.3 Class Separability Results for `Helix` Dataset Trial 1 . . . 20
5.4 Experiment 2: `SwissRoll` Dataset Normalized . . . 21
5.5 Structure Preservation Results for `SwissRoll` Dataset Trial 1 . . . 22
5.6 Class Separability Results for `SwissRoll` Dataset Trial 1 . . . 23
5.7 Structure Preservation Results for `SwissRoll` Dataset Trial 2 . . . 24
5.8 Class Separability Results for `SwissRoll` Dataset Trial 2 . . . 25
5.9 Structure Preservation Results for `SwissRoll` Dataset Trial 3 . . . 26
5.10 Class Separability Results for `SwissRoll` Dataset Trial 3 . . . 27
5.11 Structure Preservation Results for `SwissRoll` Dataset Trial 4 . . . 28
5.12 Class Separability Results for `Helix` Dataset Trial 4 . . . 29
5.13 Structure Preservation Results for `Toroid` Dataset Trial 1 . . . 30
5.14 Class Separability Results for `Toroid` Dataset Trial 1 . . . 31

B.1 Screenshot of the MATLAB live script created. . . . 37

Chapter 1

Introduction

Today, companies and organizations are relying on the use of data, and one of the important tasks they face is its collection. Data come in very large and diverse data sets whether structured, semi-structured, or unstructured, from different sources (e.g. Hyperspectral Imagery, Internet Portals, Financial tick-by-tick data, and DNA Microarrays ...), and in different sizes from terabytes to zettabytes.

In traditional data analysis [1], the data is considered to be a set of observations of such a phenomenon, these observations being a vector of values measures on various variables (e.g. blood pressure, weight, height, and among others). Traditionally, we assume many observations and a few, well-chosen variables. In recent years, we started collecting more observations but even more, to radically large numbers of variables - collecting hyper-informative detail about each observed instance. These details could be curves, media (images, videos, text...) or even spectral data so that each instance is described by thousands or billions of variables while there are only hundreds of observations.

On the other hand, this explosive growth of sample size and dimensionality of the observation vector introduces unique computational and statistical challenges, including scalability and storage bottleneck, noise accumulation, measurement errors, and more importantly the interpretability/explainability and the information visualization of the data, these are the keys to extract useful insights and make meaningful decisions [2].

To address this later matter, it is imperative to reduce the high dimensions of the data while retaining the most important and useful information based on predefined criteria, such as structure preservation and class separability for classification purposes. This is the aim of the data reduction (DR) techniques [3]. One of the most used approaches is the one based kernels (kernel dimensionality reduction - KDR), which maps the original data onto a high dimensional space (feature space) using kernels prior to a projection on a lower-dimensional space. This approach has been preferred because it captures the nonlinear relationships between independent and dependent variables with an established solution, i.e., the most dominant eigenvectors of the kernel matrix. However, since the feature space maps the original features nonlinearly, it is no longer interpretable [4].

In this work, an exhaustive overview of the fundamentals of kernel theory applied to dimensionality reduction is presented along with some experiments on SWISS Roll, Helix, and Toroidal datasets.

Chapter 2

Background

2.1 High-dimensional Data

Over the last few decades, data, data management, data processing, and data gathering have become ubiquitous factors in modern life and work. Today, industries and academia are using state-of-the-art techniques that collect massive amounts of data every moment. These data are generated from online transactions, emails, videos, audios, images, click-streams, logs, posts, search queries, health records, social networking interactions, science data, sensors, and mobile phones and their applications. The trend now is that the number of samples at the scale of hundreds and the number of variables is at the scale of billions ($p > N$). Examples include:

- Many genes, relatively few patients with a given genetic disease [5].
- Many samples of a persons' speech, relatively few speakers sampled [6].

All this amount of data is stored in databases that grow massively and become difficult to capture, form, store, manage, share, analyze and visualize via typical database software tools.

2.1.1 Big Data

Big Data [1] is a term for massive data sets having large, more varied, and complex structures with the difficulties of storing, analyzing, and visualizing for further processes or results.

Big Data is characterized by its three main components: variety, velocity, and volume as shown in Figure 2.1. Each component represented in the figure is shortly expressed below:

- *Variety* makes Big Data really big. it comes from a great variety of sources and generally has three types: structured, semi-structured, and unstructured. Structured data inserts a data warehouse already tagged and easily sorted but unstructured data is random and difficult to analyze. Semi-structured data does not conform to fixed fields but contains tags to separate data elements.

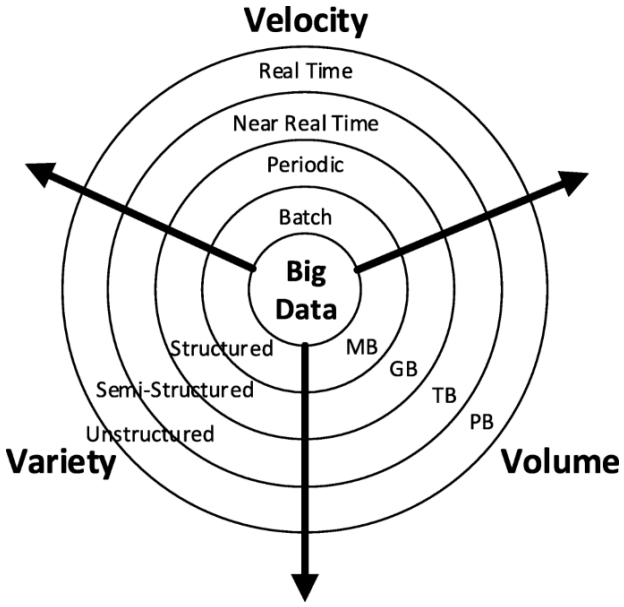

Figure 2.1: *3 Vs of Big Data.*

- *Volume* or the size of data now is larger than terabytes and petabytes. The grand scale and rise of data outstrip traditional store and analysis techniques.
- *Velocity* is required not only for Big Data but also for all processes. For time-limited processes, Big Data should be used as it streams into the organization in order to maximize its value.

Additionally, Big Data can be characterized by adding more components such as veracity, variability, and value.

2.1.2 Today's Challenges

Analyzing Big Data can require hundreds of servers running massively parallel systems. What actually distinguishes Big Data, aside from its Vs, is the potential to analyze large amounts of various data types in real-time to reveal new insights and to facilitate quick and efficient decision-making. The following describes the common today's challenges in Big Data analysis [7].

- *Heterogeneity*: Data mining algorithms locate unknown patterns and homogeneous formats for analysis in structured formats. However, the analysis of unstructured and/or semi-structured formats remains complicated. Therefore, data must be carefully structured prior to analysis. The increasing number of different data sources have produced much Big Data, both varied and heterogeneous.
- *Scalability*: Challenging issues in data analysis include the management and analysis of large amounts of data and the rapid increase in the size of datasets. Such challenges are mitigated by enhancing processor speed. However, data volume increases at a faster rate than computing resources and CPU speeds. As a result, Big Data analysis

necessitates tremendously time-consuming navigation through a gigantic search space to provide guidelines.

- *Accuracy*: Data analysis is typically buoyed by relatively accurate data obtained from structured databases with limited sources. Therefore, such analysis results are accurate. However, analysis is adversely affected by the increase in the amount of and the variety in data sources with data volume. In data stream scenarios, high-speed data strongly constrain processing algorithms spatially and temporally. Hence, stream-specific requirements must be fulfilled to process these data.

- *Complexity*: Data complexity and volume are Big Data challenges and are induced by the generation of new data (images, video, and text) from novel sources, such as smartphones, tablets, and social media networks. Thus, the extraction of valuable data is a critical issue. Additionally, validating all of the items in Big Data is almost impractical. Hence, new approaches to data qualification and validation must be introduced.

The complexity of Big Data requires other techniques that are able to reduce it into a reasonable size for processing and delivering purposes. These techniques are called Data Reduction methods, and it is discussed in the next section.

2.2 Dimensionality Reduction

In general, Data Reduction methods [3] are either used to optimize the storage or in-network movement of data or reduce data redundancy and duplication. Some of the methods only reduce the volume by compressing the original data and some of the methods reduce the velocity of data streams at the earliest before entering into big data storage systems. Alternatively, some of the methods extract topological structures of unstructured data and reduce the overall big data.

For the purpose of this work, we will focus on high-dimensions reduction, which is reducing the number of features, such a well-known method is Principal Component Analysis (PCA). Within this scope, DR approaches can be defined as the process of reducing, based on some criteria, the high dimensions of the original data (the number of variables) without losing important and useful information in order to facilitate the data analysis still to be done, such as intelligible data visualization. In the next sections, we will examine the two most important criteria to be preserved while doing DR. Class-separability ensuring and data structure preservation.

2.2.1 Class Separability Criterion

In classification problems (binary or multiple) data points are grouped in classes. Whether the main objective of the data reduction is visualization, classification, or any other downstream analysis, the common goal is to find the mapping (transformation to low-dim space) that for a given reduction in space dimension provides the maximum class-separability. In other words, we are searching among all possible transformations for the best subspace which preserves class-separations as much as possible in the lowest possible dimensional

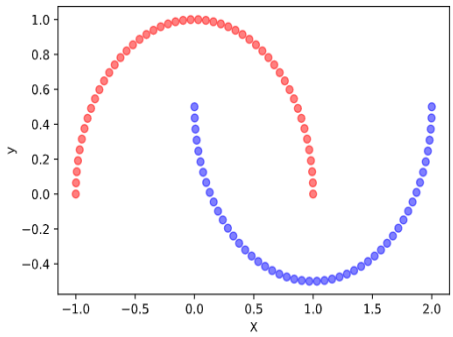

Figure 2.2: *Half Moon original dataset (2D)*

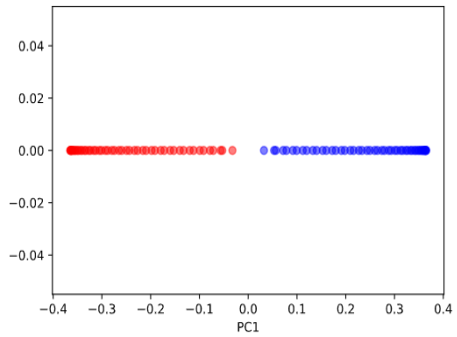

Figure 2.3: *Half Moon reduced dataset (1D)*

space. An example of DR for separated-classes preservation [8] using K-PCA is provided in Figures 2.2 and 2.3. To preserve class separation information several linear and nonlinear methods are being developed. Two of them, Linear Discriminant Analysis (LDA) and t-distributed stochastic neighbor embedding (t-SNE), are chosen to be discussed next.

- *LDA: Fisher's Linear Discriminant* [9]: In this writing we are considering LDA for a binary problem (two classes). In brief, LDA focuses on maximizing the separability among two known groups by creating a new linear axis and projecting the data points on that axis. Subsequently, a measure of separation between the projections is defined as the following :

  Let $\boldsymbol{\mu}_i$ and $\tilde{\boldsymbol{\mu}}_i$ be the mean vectors of original and projected data of the i-th class (C_i), respectively.

$$\boldsymbol{\mu_i} = \frac{1}{N_i} \sum_{\boldsymbol{x} \in Ci} \boldsymbol{x}$$

$$\text{and} \quad \tilde{\boldsymbol{\mu}}_{\boldsymbol{i}} = \frac{1}{N_i} \sum_{\boldsymbol{x} \in Ci} \boldsymbol{w^T x} = \boldsymbol{w^T} \frac{1}{N_i} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x} = \boldsymbol{w^T \mu_i}$$

  where $\boldsymbol{w}$ is the projection vector.

  For two classes, the L2 norm objective function is defined as:

$$\boldsymbol{J(w)} = \|\tilde{\boldsymbol{\mu}}_1 - \tilde{\boldsymbol{\mu}}_2\|_{L_2} = \left\|\boldsymbol{w^T \mu_1} - \boldsymbol{w^T \mu_2}\right\|_{L_2} = \left\|\boldsymbol{w^T}(\boldsymbol{\mu_1} - \boldsymbol{\mu_2})\right\|_{L_2}$$

  The idea behind Fisher's LD is to maximize the variability among classes (the distance between the means) and to minimize the invariance within the class. This latter is defined as the sum of squared differences between the projected data points and their corresponded class mean.

$$\tilde{\boldsymbol{S}}_i^2 = \sum_{\boldsymbol{x} \in C_i} (\boldsymbol{w^T x} - \tilde{\boldsymbol{\mu}}_i)^2$$

$\tilde{\boldsymbol{S}}_1^2 + \tilde{\boldsymbol{S}}_2^2$ measures the variability within the two classes.

Then the objective function to be maximized becomes:

$$\boldsymbol{J}(\boldsymbol{w}) = \frac{\left\|\boldsymbol{w}^{\boldsymbol{T}}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right\|^2}{\tilde{\boldsymbol{S}}_1^2 + \tilde{\boldsymbol{S}}_2^2}$$

Therefore, we will be looking for a projection where samples from the same class are projected very close to each other and, at the same time, the projected means are as farther apart as possible.

LDA can be used for multiple classes as well, but the data points could only be projected onto C-1 (C number of classes) dimensions.
In order to extract nonlinear features Kernel Discriminant Analysis (KDA) could be used, it uses the same kernel trick as in K-PCA (more details in the coming sections).

- *t-SNE* [10]: it is a non-linear technique for DR that is particularly well suited for the visualization of high-dimensional datasets.

  Here is a brief overview of the working of t-SNE:

  – The algorithm starts by calculating the probability of similarity of points in both high-dimensional and the corresponding low-dimensional spaces.
  The similarity of data point $\boldsymbol{x}_j$ to data point $\boldsymbol{x}_i$ is the conditional probability, $p_{j|i}$, that $\boldsymbol{x}_i$ would pick $\boldsymbol{x}_j$ as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian Distribution centered at $\boldsymbol{x}_i$. For nearby data points, $p_{j|i}$ is relatively high, whereas for widely separated data points, $p_{j|i}$ will be almost infinitesimal. Mathematically, the conditional probability $p_{j|i}$ is given by:

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

  Where σ_i is the variance of the Gaussian that is centered on data point $\boldsymbol{x}_i$. For the low-dimensional counterparts $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ of the high-dimensional data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, it is possible to compute a similar conditional probability, which we denote by $q_{j|i}$ as:

  $$q_{j|i} = \frac{\exp(-\left\|\boldsymbol{y}_i - \boldsymbol{y}_j\right\|^2)}{\sum_{k \neq i} \exp(-\left\|\boldsymbol{y}_i - \boldsymbol{y}_k\right\|^2)}$$

  Where the variance, σ_i, of Gaussian is set to $\frac{1}{\sqrt{2}}$. In both spaces (high and low dims) we set $p_{i|i}$ and $q_{i|i}$ to zero because we are only interested in modeling pairwise similarities.
  If the map points $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ correctly model the similarity between the high-dimensional data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, the conditional probabilities $p_{j|i}$ and $q_{j|i}$ will be equal.

– Then, it tries to find a low-dimensional data representation that minimizes the mismatch between these conditional probabilities (similarities) $p_{j|i}$ and $q_{j|i}$. t-SNE minimizes the sum of Kullback-Leibler divergences over all data points using a gradient descent method. The cost function is given by:

$$C = \sum_i KL(P_i \| Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

in which P_i represents the conditional probability distribution over all other data points given data point $\boldsymbol{x}_i$, and Q_i represents the conditional probability distribution over all other map points given map point $\boldsymbol{y}_i$.

In this way, t-SNE maps the multi-dimensional data to a lower dimensional space and attempts to find patterns in the data by identifying observed clusters based on similarity of data points with multiple features. However, after this process, the input features are no longer identifiable, and it is not possible to make any inference based only on the output of t-SNE. Hence, it is mainly used for data exploration and visualization purposes.

2.2.2 Structure Preservation

Another objective of data reduction methods is to reduce the data dimensions while preserving its geometrical and topological features. For this purpose, a new approach named manifold learning (ML) [11] has been introduced recently. A manifold, simply put, is a surface of any shape, not necessarily in only two dimensions but it could be generalized to n-dimensions, and it's no longer called a plane or surface but rather a manifold.

The manifold learning algorithms can be viewed as the non-linear version of PCA. PCA tries to project the data onto some low-dimensional surface. But this is restrictive in the sense that those surfaces are all linear. What if the best representation lies in some weirdly shaped surface? PCA will miss that. For example, the Swiss Roll dataset in Figure 2.4 holds no linear underlying surface, so that PCA could give accurate results. In fact, PCA will search for a planar surface to describe this data, that doesn't exist. Fortunately, manifold learning solves this problem very efficiently.

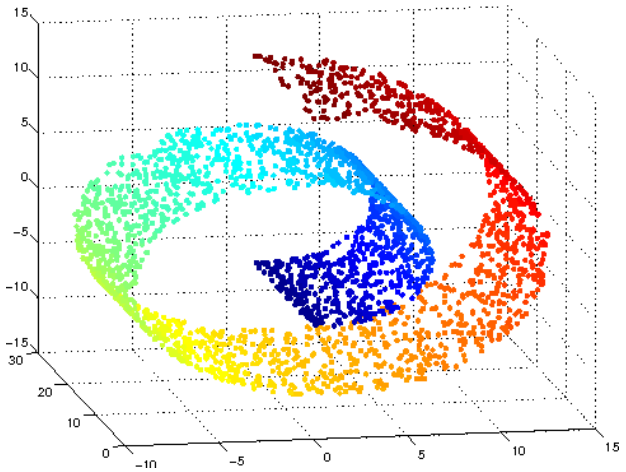

Figure 2.4: *A curled plane: the Swiss Roll.*

The idea of manifold learning methods such as Locally Linear Embedding, Isomap, Spectral Embedding, and more, is that the dimensionality of many data sets is only artificially high. Although the data points may consist of thousands or even millions of features, they may be described as a function of only a few underlying parameters. That is, the data points are actually samples from a low-dimensional manifold that is embedded in a high-dimensional space. The goal of manifold learning methods is to uncover these parameters in order to find a low-dimensional representation of the data.

Chapter 3

Overview on Kernels for Dimensionality Reduction

3.1 Basics on Dimensionality Reduction

3.1.1 Spectral Analysis

In data analysis, the problem of dealing with high dimensional data and the need of using DR methods is often presented. In a precise mathematical framework, DR could be defined as if we are given a set of high dimensional data $x_1, x_2, ..., x_n$ in R^d (inputs), and we need to compute their representations $y_1, y_2, ..., y_n$ in R^r (outputs), with $r << d$. It is usually assumed that the inputs were sampled from a low-dimensional manifold embedded in R^d. Ideally, an algorithm should be able to estimate the value of r that is required for a low-dimensional representation as well to compute the low dimensional representations. If the original data are confined to a linear subspace, then a low-dimensional representation can be easily found by classical spectral methods such as principal component analysis (PCA) and metric multidimensional scaling (MDS). For data sampled from general nonlinear manifolds, however, these linear methods do not give satisfactory results, and nonlinear spectral or non-spectral methods such as Isomap, locally linear embedding (LLE) and Kernel-PCA are needed [12].

In the context of DR, these spectral methods rely on spectral analysis which is, in brief, the process of finding eigenvalues and eigenvectors of a given matrix, what is called eigen-decomposition. For example, The eigenvalue decomposition of the covariance matrix for PCA or the Gram matrix for MSD. These methods offer nice theoretical properties such as exact optimization, simplicity, the possibility to build embeddings incrementally (in PCA adding orthogonal dimensions as needed), and more. However, they do not prove to be very robust against departures from their underlying model. For instance, Isomap does not produce satisfying embeddings for non-developable manifolds. In contrast, iterative methods based on gradient descent, for example, can deal with more complex objective functions, whereas spectral methods are restricted to functions having "nice" algebraic properties. The price to pay, of course, is a heavier computational load and a larger number of parameters to be tuned.

3.1.2 Principal component analysis (PCA) from a distance-based viewpoint

PCA is a standard and popular statistical approach that tries to explain a large number of highly correlated variables (features of the data matrix) by a few components (new variables) that preserve important information (in this case variance). Final results of PCA are usually easy to interpret and make inference with original variables, because of the linear relationship between them.

Let $\boldsymbol{X}$ be an $n \times d$ matrix (n: samples and d: variables). The goal of PCA is to project the data onto a space having dimensionality $r \leq d$ while maximizing the variance (distance) of the projected data points.

To begin with, we'll consider the projection onto a one-dimensional space ($r = 1$). We can define the direction of this space using a d-dimensional vector u_1. Each data point $\boldsymbol{x}_n$ is then projected onto a scalar value :

$$\boldsymbol{u}_1^T \boldsymbol{x}_n = \sum_{i=1}^{d} \boldsymbol{u}_{1i} \boldsymbol{x}_{ni}$$

The mean of the projected data is $\boldsymbol{u}_1^T \bar{\boldsymbol{x}}$ where $\bar{\boldsymbol{x}}$ is the data points mean given by

$$\bar{\boldsymbol{x}} = \frac{\sum_{i=1}^{n} \boldsymbol{x}_i}{n}$$

and then the variance of the projected data is given by the squared distance between projected data points and the projected mean

$$\frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{u}_1^T \boldsymbol{x}_i - \boldsymbol{u}_1^T \bar{\boldsymbol{x}})^2 = \boldsymbol{u}_1^T \boldsymbol{S} \boldsymbol{u}_1$$

($\boldsymbol{u}_1$ is assumed to be a unit vector, $\boldsymbol{u}_1^T \boldsymbol{u}_1 = 1$, since we are only interested in its direction) where $\boldsymbol{S}$ is the data covariance matrix defined as

$$\boldsymbol{S} = \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T$$

Now the PCA objective function is

$$\begin{aligned} &\max_{\boldsymbol{u}} \boldsymbol{u}_1^T \boldsymbol{S} \boldsymbol{u}_1 \\ &\text{s.t.} \quad \|\boldsymbol{u}_1\| = 1 \end{aligned}$$

After introducing Lagrange multiplier (λ_1), we get the equivalent unconstrained maximization problem

$$\max_{\boldsymbol{u}} \boldsymbol{u}_1^T \boldsymbol{S} \boldsymbol{u}_1 + \lambda_1 (1 - \boldsymbol{u}_1^T \boldsymbol{u}_1)$$

By setting the derivative with respect to $\boldsymbol{u}_1$ equal to zero, we see that this quantity has a stationary point when

$$\boldsymbol{S} \boldsymbol{u}_1 = \lambda_1 \boldsymbol{u}_1 \iff \boldsymbol{u}_1^T \boldsymbol{S} \boldsymbol{u}_1 = \lambda_1$$

So the variance will be a maximum when we set $\boldsymbol{u}_1$ equal to the eigenvector having the largest eigenvalue λ_1 of the covariance matrix S. This eigenvector is known as the first principal component. Additional principal components (up to d) can be defined in an incremental fashion by choosing each new direction to maximize the projected variance and to be orthogonal to those already considered.

A classical way of doing PCA is to use Singular Value Decomposition. For simplicity, we are considering that $\boldsymbol{X}$ is already centered, which is a required step in PCA.

The SVD of $\boldsymbol{X}$ is given by:

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$$

where $\boldsymbol{U} = (\boldsymbol{u}_1, ..., \boldsymbol{u}_d)$ is an $n \times d$ orthogonal matrix satisfying $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}_d$, $\boldsymbol{V} = (\boldsymbol{v}_1, ..., \boldsymbol{v}_d)$ is a $d \times d$ orthogonal matrix satisfying $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}_d$, and $\boldsymbol{D} = diag(d_1, ..., d_d)$ is a $d \times d$ diagonal matrix holding singular values. The singular values are assumed to be ordered such that $d_1 \geq d_2 \geq ... \geq d_d \geq 0$. The columns of $\boldsymbol{U}\boldsymbol{D}$ are the principal components (PCs) and the columns of $\boldsymbol{V}$ are the corresponding loadings. The j-th PC is $PC_j = d_j\boldsymbol{u_j}$ and its sample variance is d_j^2/n.
For any integer $r \leq d$, let

$$\boldsymbol{X}_r = \sum_{j=1}^{r} d_j \boldsymbol{u}_j \boldsymbol{v}_j^T = \boldsymbol{U}_r \boldsymbol{D}_r \boldsymbol{V}_r^T$$

where $\boldsymbol{U}_r = (\boldsymbol{u_1}, ..., \boldsymbol{u}_r)$, $\boldsymbol{D}_r = diag(d_1, ..., d_r)$, and $\boldsymbol{V}_r = (\boldsymbol{v}_1, ..., \boldsymbol{v}_r)$. The variation of $\boldsymbol{X}_r$ is $\sum_{j=1}^{r} d_j^2/n$ and its proportion to the total variation is

$$\lambda_r = \frac{\sum_{j=1}^{r} d_j^2}{\sum_{j=1}^{d} d_j^2}$$

If there is a small r such that $\lambda_r \approx 1$, then the dimension of the data matrix is successfully reduced from d to r with most variations contained in $\boldsymbol{X}_r$. The reduced data matrix is $\boldsymbol{U}_r\boldsymbol{D}_r = (PC_1, ..., PC_r)$.

3.2 Kernel Functions

3.2.1 The Concept of Kernel

The similarity measure, in input or output space, is the core of learning theory, therefore it should be chosen accordingly. Let's consider a similarity measure of the form:

$$\begin{aligned} k \colon \boldsymbol{X} \times \boldsymbol{X} &\to \mathbb{R} \\ (\boldsymbol{x}, \boldsymbol{x}') &\mapsto k(\boldsymbol{x}, \boldsymbol{x}') \end{aligned}$$

that is, a function that, given two data points (vectors) $\boldsymbol{x}$ and $\boldsymbol{x}'$ in $\boldsymbol{X}$, returns a real number characterizing their similarity. This function k is called **kernel**. With the properties:

- It's assumed that kernel k is symmetric, that is,

$$k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}', \boldsymbol{x}) \quad \forall \boldsymbol{x}, \boldsymbol{x}' \in \boldsymbol{X}$$

- kernel k is positive semi-definite (PSD)

$$\forall c_i \in \mathbb{R}, \boldsymbol{x}_i \in \boldsymbol{X}, \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$$

- Kernels like measures are (often) non-negative:

$$k(\boldsymbol{x}, \boldsymbol{x}') \geq 0$$

- Kernels are expressing similarity measurement:

$$[k(\boldsymbol{x}, \boldsymbol{x}') \leq 1 \quad \text{and} \quad k(\boldsymbol{x}, \boldsymbol{x}') = 1] \Longleftrightarrow \boldsymbol{x} = \boldsymbol{x}'$$

3.2.2 Kernel matrix

In general, a kernel matrix (or Gram matrix) of a set of vectors (samples in high-dims) $(\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}) \in \boldsymbol{X}$ is defned to be a $n \times n$ matrix $\boldsymbol{K}$ as:

$$\boldsymbol{K} = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \dots & k(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1)\rangle & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \dots & k(\boldsymbol{x}_2, \boldsymbol{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1) & k(\boldsymbol{x}_n, \boldsymbol{x}_2) & \dots & k(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix}$$

For a well-defined kernel function k.

3.2.3 Kernel Trick

Kernel-based methods sometimes need to transform the input data into some high-dimensional space, like in Kernel Principal Component Analysis (K-PCA). This mapping is done by a function, $\phi : \mathbb{X} \longrightarrow \mathbb{H}$. This high-dimensional mapping can seriously increase computation time, to get around this problem the kernel trick is used: $k(\boldsymbol{x}, \boldsymbol{x\prime}) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x\prime})$, it only calculates the inner product between data points without even defining the mapping ϕ explicitly.
Given any algorithm that can be expressed solely in terms of dot products, this trick allows us to construct different nonlinear versions of it.

3.2.4 Types of Kernel

Here is a list of common kernels: Let $\boldsymbol{x}$ and $\boldsymbol{x}'$ represent the feature vectors in input space $\boldsymbol{X}$

- *Polynomial Kernel*: is useful for non-linear linear problems, it takes into account the interaction features. For degree-p polynomials, it is defined as

$$k(\boldsymbol{x}, \boldsymbol{x}') = (c + \boldsymbol{x}^T \boldsymbol{x}')^p$$

  where $c \geq 0$ is a free parameter representing the influence of higher-order versus lower-order terms in the polynomial.

- *Radial Basis Function (RBF)/Gaussian Kernel*: the most used kernel in machine learning. In particular, it is commonly used in support vector machine classification (SVM)

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\gamma \left\| \boldsymbol{x} - \boldsymbol{x}' \right\|^2\right)$$

  where $\gamma = \frac{1}{2\sigma^2}$, and σ a free variable.

- *Linear Kernel*: is just the $dot - product$

$$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle = \boldsymbol{x}^T \boldsymbol{x}'$$

3.3 Kernel PCA

3.3.1 Feature Space (High-dimensional Space)

The goal of dimensionality reduction is to embed a high-dimensional data matrix of n observations $\boldsymbol{x}_i \in \boldsymbol{X}^d$ with $i \in \{1, ...n\}$ into a low-dimensional space of r dimensions ($r \leq d$) [11]. Particularly, PCA tries to transfer observations from input space into a low-dimensional space while preserving the maximum variation among variables (features). This variance could be seen as a distance that PCA maximizes. Such a distance could be the (canonical) $dot - product$, defined as:

$$\langle \boldsymbol{x}, \boldsymbol{x}' \rangle := \sum_{i=1}^{n} x_i x_i' \quad \text{For } \boldsymbol{x} \text{ and } \boldsymbol{x}' \in \mathbb{R}^n.$$

The $dot - product$ is very useful because it carries out all the geometric construction that can be formulated in terms of angles, lengths ($\|x\|$), and distances ($\|x - x'\|$). However, we must assume that the input space defines it in the first place which is not always the case when observations are some kind of object, such as "strings".

In this case, we therefore first need to represent (map) the input data as vectors in some $dot - product$ space $\mathbb{H}$:

$$\begin{aligned} \boldsymbol{\phi} &: \boldsymbol{X} \to \mathbb{H} \\ \boldsymbol{x} &\mapsto \tilde{\boldsymbol{x}} := \boldsymbol{\phi}(\boldsymbol{x}) \end{aligned}$$

Even if the input data exist in a $dot - product$ space, we may still consider applying that map. In that case, $\boldsymbol{\phi}$ will typically be a nonlinear map.
This new space $\mathbb{H}$ is the feature (Hilbert) [13] space that could be of any dimensions, even infinite dimensions.
The important key feature of this transformation is the ability to map a non-linear input space to a linear one in a new feature space. As in figures B.1 and 3.2. An example of a binary classification problem mapped into feature space. We assume that the true decision boundary is an ellipse in input space (figure B.1). The task is to estimate this non-linear boundary decision. When mapped into feature space (Figure 3.2) via the nonlinear map,

$$\phi(\boldsymbol{x}) = (z_1, z_2, z_3) = (x_1^2, x_2^2, \sqrt{2} x_1 x_2) \quad \forall \boldsymbol{x} \in \mathbb{R}^3$$

the ellipse becomes a hyperplane.

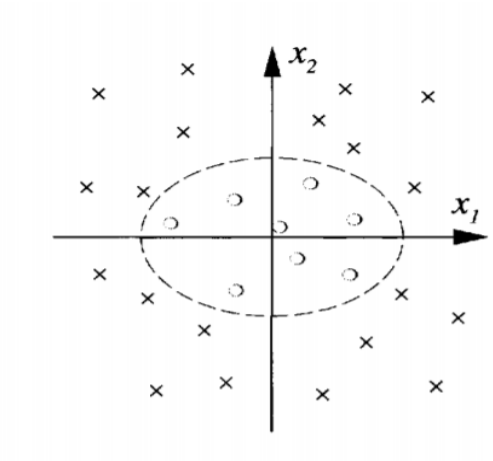

Figure 3.1: *Non-linear binary classification problem in 2D.*

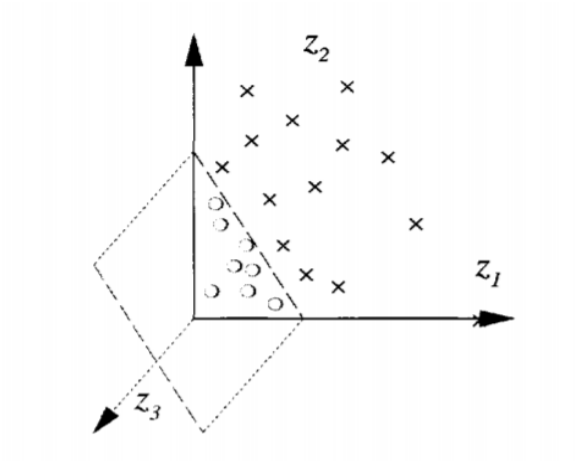

Figure 3.2: *Linearly-separable classification problem in higher dimensions (3D).*

3.3.2 Kernel PCA from Distance-based Viewpoint

The K-PCA extends the classical PCA to a high dimensional feature space $\mathbf{H}$ using a mapping ϕ. It can extract up no n (number of observations) non-linear principal components without explicitly calculating the mapping itself. In other words, K-PCA is just PCA, but instead of extracting the principal components in input space, they are extracted from the feature space[8].
Let $k : \mathbf{R}^d \times \mathbf{R}^d \longrightarrow \mathbf{R}$ with the property that there exist a map, $\phi : \mathbf{R}^d \longrightarrow \mathbf{H}$, into a dot product feature space $\mathbf{H}$ such that:

$$\forall \boldsymbol{x}, \boldsymbol{x\prime} \in \mathbf{R}^d \quad k(\boldsymbol{x}, \boldsymbol{x\prime}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x\prime}) \rangle = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x\prime})$$

For the simplicity, we suppose that the mean of the data in feature space $\mathbf{H}$ is zero

$$\mu = \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i) = 0$$

Then, the covariance matrix in feature space is defined as:

$$\boldsymbol{C} = \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i) \phi(\boldsymbol{x}_i)^T$$

As in PCA, the goal is to solve this eigen-problem:

$$\begin{aligned} \boldsymbol{C}\boldsymbol{v} &= \lambda \boldsymbol{v} \\ \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i) \phi(\boldsymbol{x}_i)^T \boldsymbol{v} &= \lambda \boldsymbol{v} \\ \boldsymbol{v} &= \frac{1}{n\lambda} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i) \phi(\boldsymbol{x}_i)^T \boldsymbol{v} \\ \boldsymbol{v} &= \frac{1}{n\lambda} \sum_{i=1}^{n} \langle \phi(\boldsymbol{x}_i), \boldsymbol{v} \rangle \phi(\boldsymbol{x}_i)^T \end{aligned}$$

prof of $\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{v} = (\langle \boldsymbol{x}\boldsymbol{v} \rangle)\boldsymbol{x}^T$ in appendix.
We know that a eigenvector $\boldsymbol{v_j}$, associated with $\lambda_j \neq 0$, is a linear combination of features of space $\mathbf{H}$:

$$\boldsymbol{v_j} = \sum_{i=1}^{n} \alpha_i^j \phi(\boldsymbol{x}_i)$$

α_i^j means the i-th value of j-th vector.
So, Finding a eigenvector is equivalent to finding the coefficients of α^j. By substituting $\boldsymbol{v_j}$ back into formula above, we get:

$$\frac{1}{n}\sum_{i=1}^{n} \phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_i)^T(\sum_{l=1}^{n} \alpha_l^j \phi(\boldsymbol{x}_l)) = \lambda_j \sum_{l=1}^{n} \alpha_l^j \phi(\boldsymbol{x}_l)$$
$$\frac{1}{n}\sum_{i=1}^{n} \phi(\boldsymbol{x}_i)(\sum_{l=1}^{n} \alpha_l^j \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_l)) = \lambda_j \sum_{l=1}^{n} \alpha_l^j \phi(\boldsymbol{x}_l)$$

Multiply by $\phi(\boldsymbol{x}_p)^T$ for $p \in 1, 2, ...N$:

$$\frac{1}{n}\sum_{i=1}^{n} \phi(\boldsymbol{x}_p)^T\phi(\boldsymbol{x}_i)(\sum_{l=1}^{n} \alpha_l^j \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_l)) = \lambda_j \sum_{l=1}^{n} \alpha_l^j \phi(\boldsymbol{x}_p)^T\phi(\boldsymbol{x}_l)$$
$$\frac{1}{n}\sum_{i=1}^{n} k(\boldsymbol{x}_p, \boldsymbol{x}_i)(\sum_{l=1}^{n} \alpha_l^j k(\boldsymbol{x}_i, \boldsymbol{x}_l)) = \lambda_j \sum_{l=1}^{n} \alpha_l^j k(\boldsymbol{x}_p, \boldsymbol{x}_l)$$

By plugging in the kernel matrix described in the previous section we get:

$$\boldsymbol{K}^2\boldsymbol{\alpha^j} = n\lambda_j \boldsymbol{K}\boldsymbol{\alpha^j}$$
$$\boldsymbol{K}\boldsymbol{\alpha^j} = n\lambda_j \boldsymbol{\alpha^j}$$

In fact, removing $\boldsymbol{K}$ will affect eigenvectors with zero eigenvalues. Fortunately, they are not considered as principal components.
The normalization constraint:

$$\boldsymbol{v}_j^T\boldsymbol{v}_j = 1 \tag{3.1}$$
$$\sum_{p=1}^{n}\sum_{l=1}^{n} \alpha_l^j \alpha_p^j \phi(\boldsymbol{x}_l)^T\phi(\boldsymbol{x}_p) = 1 \tag{3.2}$$
$$\boldsymbol{\alpha^j}^T\boldsymbol{K}\boldsymbol{\alpha}^j = 1 \tag{3.3}$$
$$n\lambda_j\boldsymbol{\alpha^j}^T\boldsymbol{\alpha}^j = 1 \tag{3.4}$$

Projection of $\boldsymbol{x}$ onto j-th principal component is given by:

$$\phi(\boldsymbol{x})^T\boldsymbol{v}_j = \sum_{i=1}^{n} \alpha_i^j \phi(\boldsymbol{x})^T\phi(\boldsymbol{x}_i) = \sum_{i=1}^{n} \alpha_i^j k(\boldsymbol{x}, \boldsymbol{x}_i)$$

In global, K-PCA steps:

- Kernel choice
- Normalization (when data are not centered)
- Solving the eigenvalue problem: $\boldsymbol{K}\alpha^j = \lambda_j\alpha^j$
- Project (new or old) data points as: $\boldsymbol{y}_i = \sum_{i=1}^{n} \alpha_i^j k(\boldsymbol{x}, \boldsymbol{x}_i), \quad i \in 1, ..$

Chapter 4

Experimental Setup

Dimensionality reduction is an interactive process, meaning there are several parameters to tun to find a better representation of the original data or a very accurate classifier. These parameters differ from one kernel method to another. For example, the polynomial kernel uses the degree of the polynomial as a parameter, each degree gives varying results that could be interpreted differently depending on the nature of the problem at hand. In order to better understand DR approaches and especially the famous one K-PCA, several experiments were run using two kernel functions, Gaussian and Polynomial, along with a variety of parameters.

The workflow consists of four main parts (Data, Kernel functions, K-PCA, Results) as depicted in Figure 4.1.

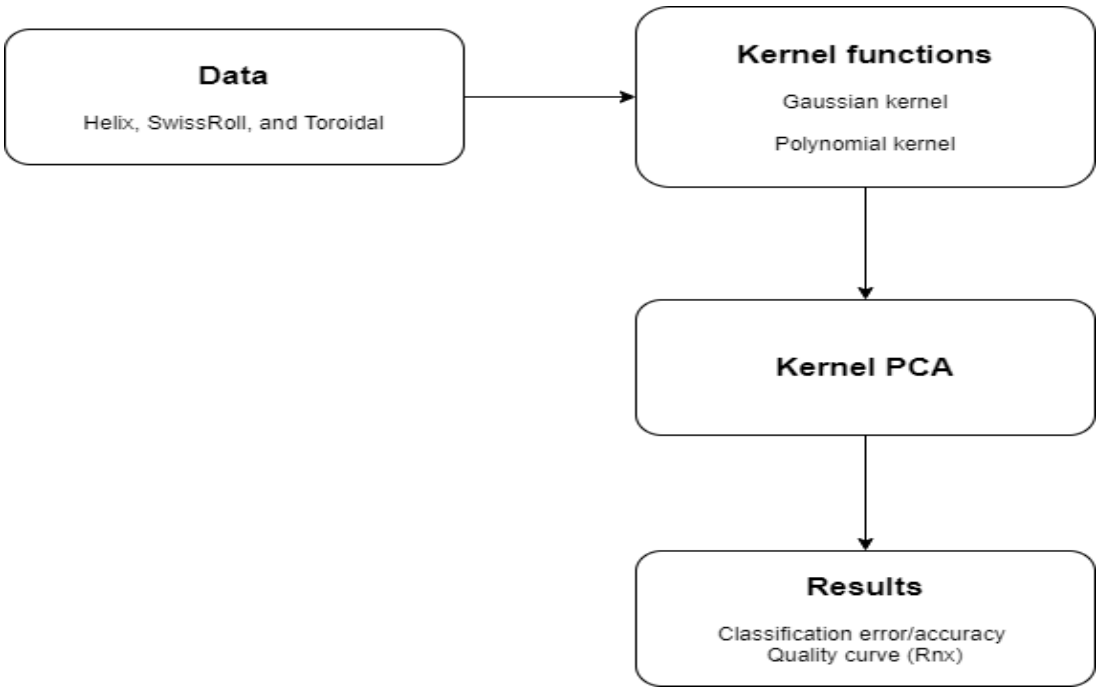

Figure 4.1: *Proposed experimental setup to compare kernel matrices on dimensionality reduction tasks using Kernel PCA.*

- *Data*: in this work, several datasets were used to demonstrate two objectives. First is the structure preservation in which the SwissRoll dataset was mainly used besides Helix and Toroidal. Second is the separability criterion using the Spherical dataset trying to embed it into another space where classes are more separable than in the original space.
- *Kernel functions* here the two famous ones were used:
  - *Gaussian kernel*:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{2\sigma^2}\left\|\boldsymbol{x} - \boldsymbol{x}'\right\|^2\right)$$

    σ a free parameter.
  - *Polynomial kernel*:

$$k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T \boldsymbol{x}')^p$$

    p the polynomial degree, a free parameter.
- *Kernel PCA*: in this last step the KPCA algorithm was used to embed the original data into a lower-dimensional space.
- *Results*: at the end the important measures as they are an important part of DR approach [8].
  - *Classification error/accuracy*: Here several classifiers, such as Linear/Quadratic Discriminant Classifier (LDC), and in most cases, parzen, were used on the embedded data in order to determine whether it is possible to find a simple decision boundary (ideally linear). Also, a decision tree and a density based classifier using Parzen distribution (ParzenC) are used.
  - *Quality curve Rnx* This performance measures the quality of the embedded data using RNX(k) [14] calculated at each value of k from 2 to n-1 presented in a graph. The area under the curve (AUC) is used for this Rnx graph to calculate the overall performance.

Chapter 5

Results and Discussion

5.1 Experiment 1 : `Helix` Dataset

In this first experiment we're using the `Helix` dataset normalized between $[-1, 1]$ presented in the Figure 5.1 below (plot title should be: Dataset-`Helix`).

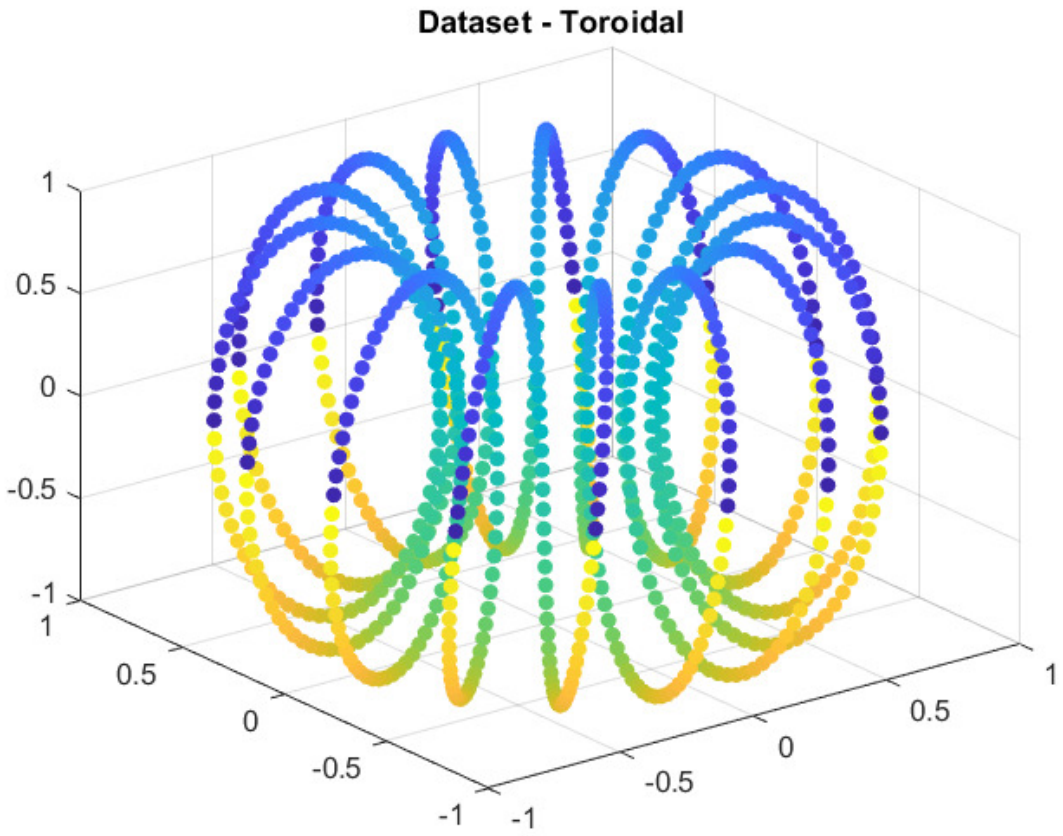

Figure 5.1: Experiment 1: `Helix` Dataset Normalized

The trial settings summary is as shown in Table 5.1.

Table 5.1: Trial settings for `Helix` dataset

Trial	Kernel function	Parameters	Classifier Accuracy
1	Polynomial	19-degree	ParzenC: 0.90

5.1.1 Structure Preservation Results

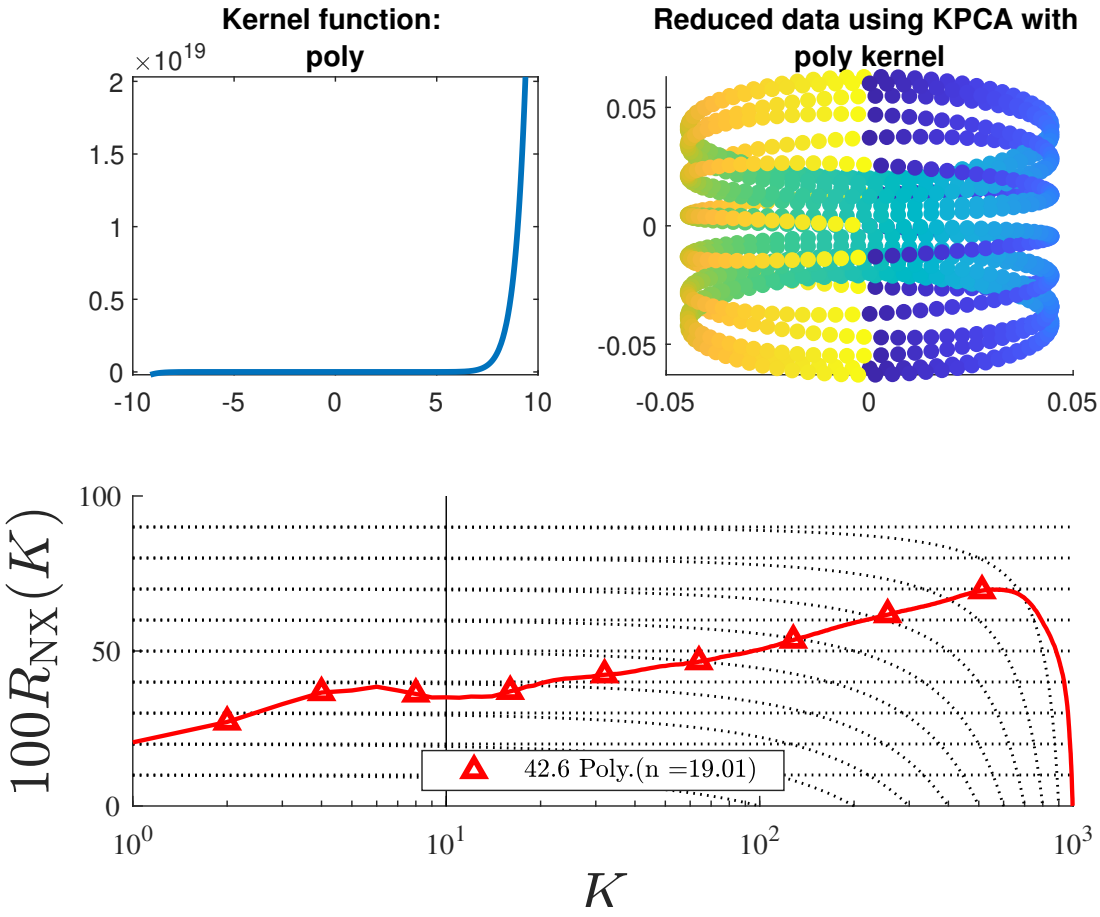

Figure 5.2: Structure Preservation Results for `Helix` Dataset Trial 1

As it can be seen from the Rnx plot, Figure 5.2, the maximum value doesn't exceed 80% for a big value of k around 10^3. The polynomial kernel did capture some global structures from the original data but it missed the important details.

5.1.2 Class Separability Results

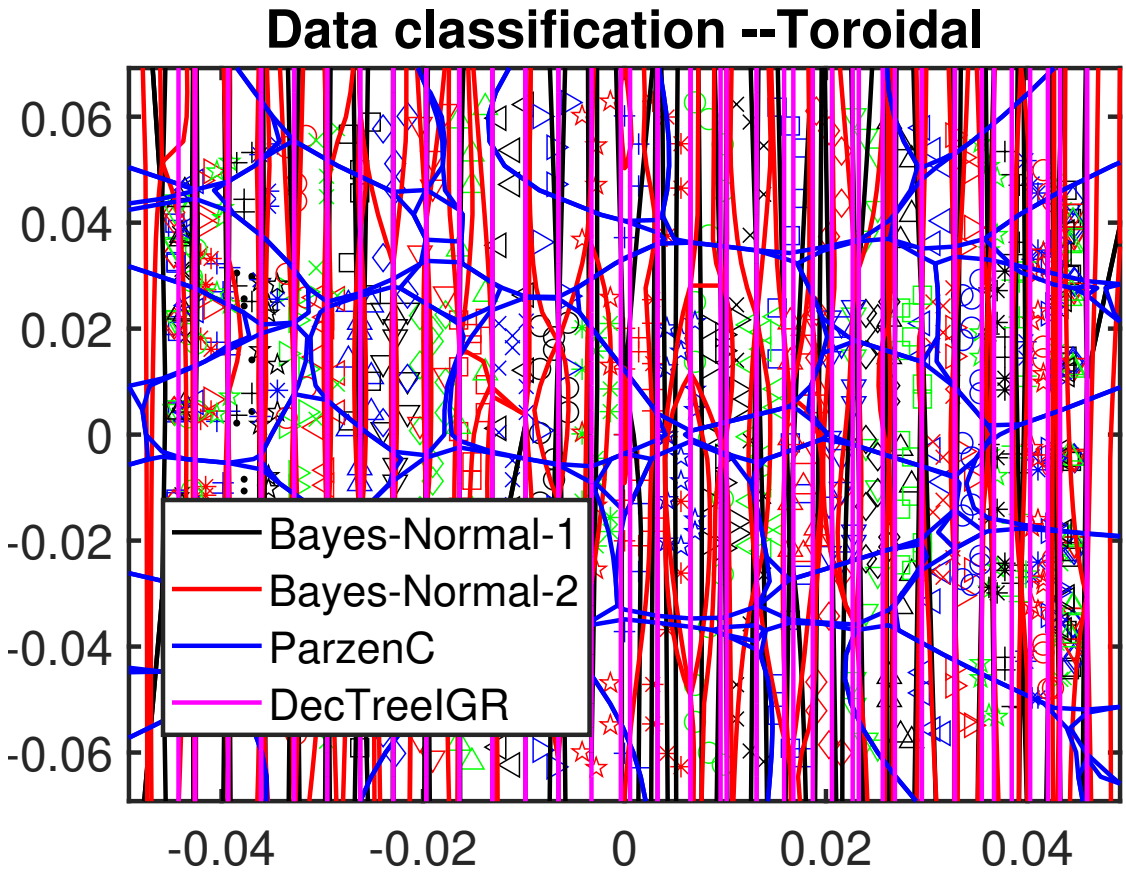

Figure 5.3: Class Separability Results for `Helix` Dataset Trial 1

The best performant classifier is ParzenC with an accuracy of 90%.

5.2 Experiment 2 : `SwissRoll` Dataset

In this experiment the `SwissRoll` dataset (normalized) is being used. Figure 5.4.

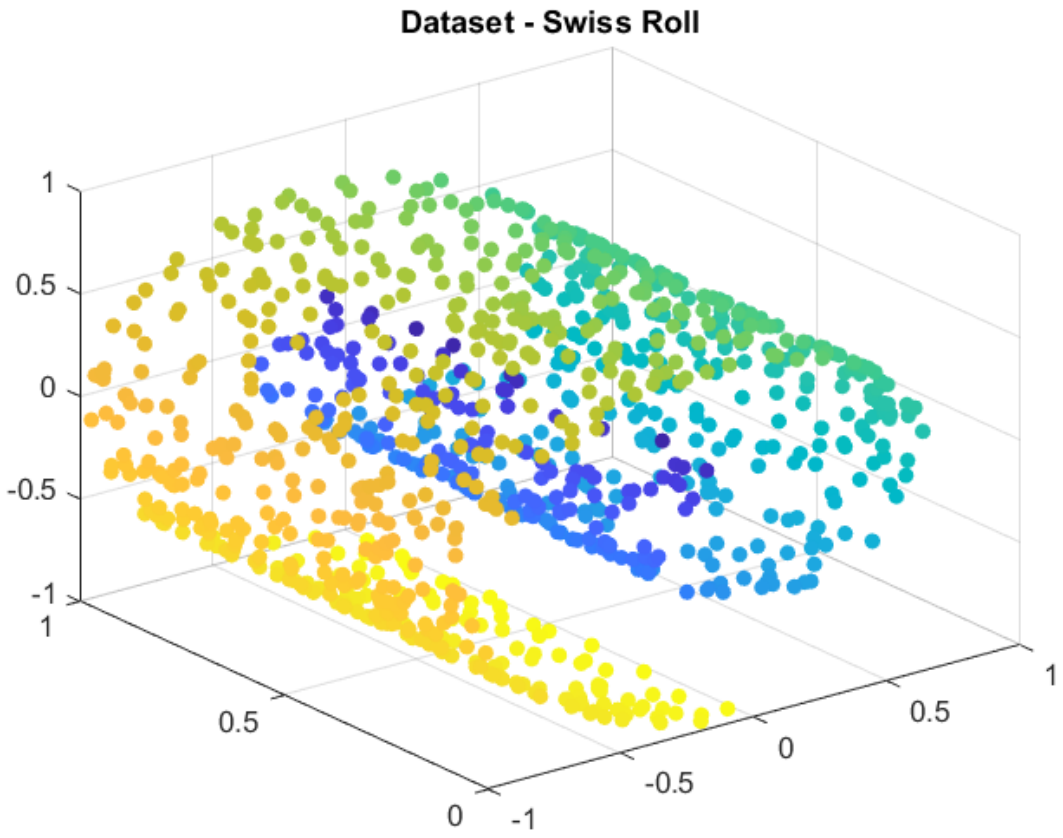

Figure 5.4: Experiment 2: `SwissRoll` Dataset Normalized

The trial settings summary is as shown in Table 5.2

Table 5.2: Trial settings for `SwissRoll` dataset

Trial	Kernel function	Parameters	Classifier Accuracy
1	Gaussian	$\sigma = 5.68$	DecisionTree: 0.9841
2	Gaussian	$\sigma = 0.73$	Bayes-Normal-2: 9669
3	Gaussian	$\sigma = 0.43$	ParzenC: 0.7824
4	Gaussian	$\sigma = 7$	ParzenC: 0.9856

5.2.1 Trial 1:

5.2.2 Structure Preservation Results

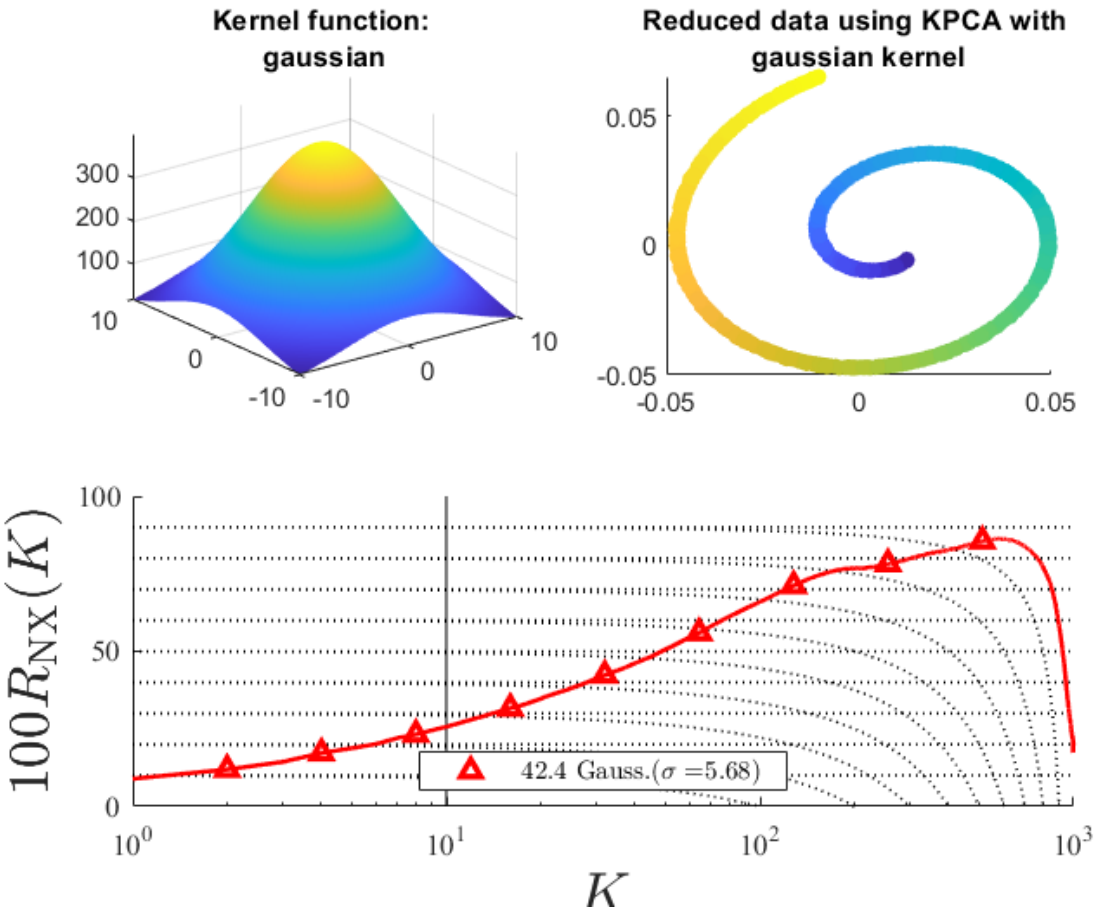

Figure 5.5: Structure Preservation Results for `SwissRoll` Dataset Trial 1

The graph in Figure 5.5 shows that the kernel function was able to capture global structures very well, when k around 10^3 it hits 100%, but for small values of k the kernel is having troubles learning these little structures.

5.2.3 Class Separability Results

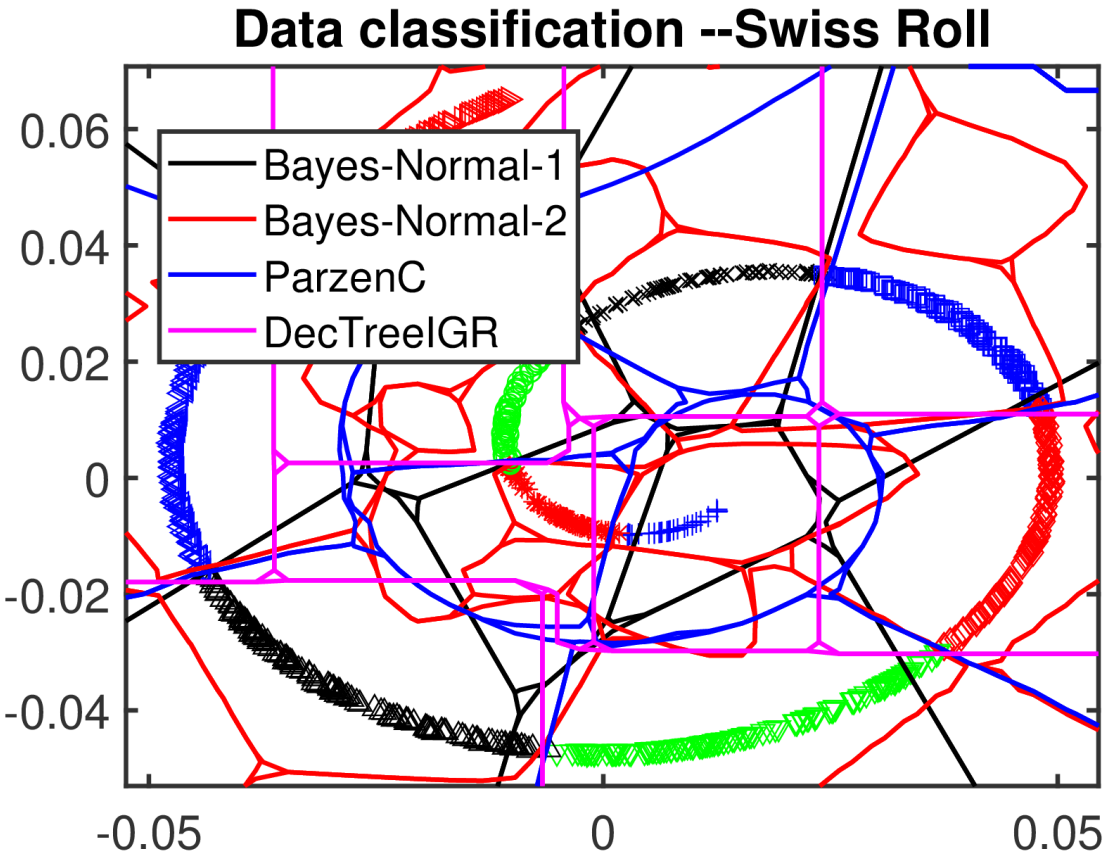

Figure 5.6: Class Separability Results for `SwissRoll` Dataset Trial 1

The best performant classifier is DecisionTree with an accuracy of 98.41%. This is due to the fact that the embedded data is non-linearly separable.

5.2.4 Trial 2:

5.2.5 Structure Preservation Results

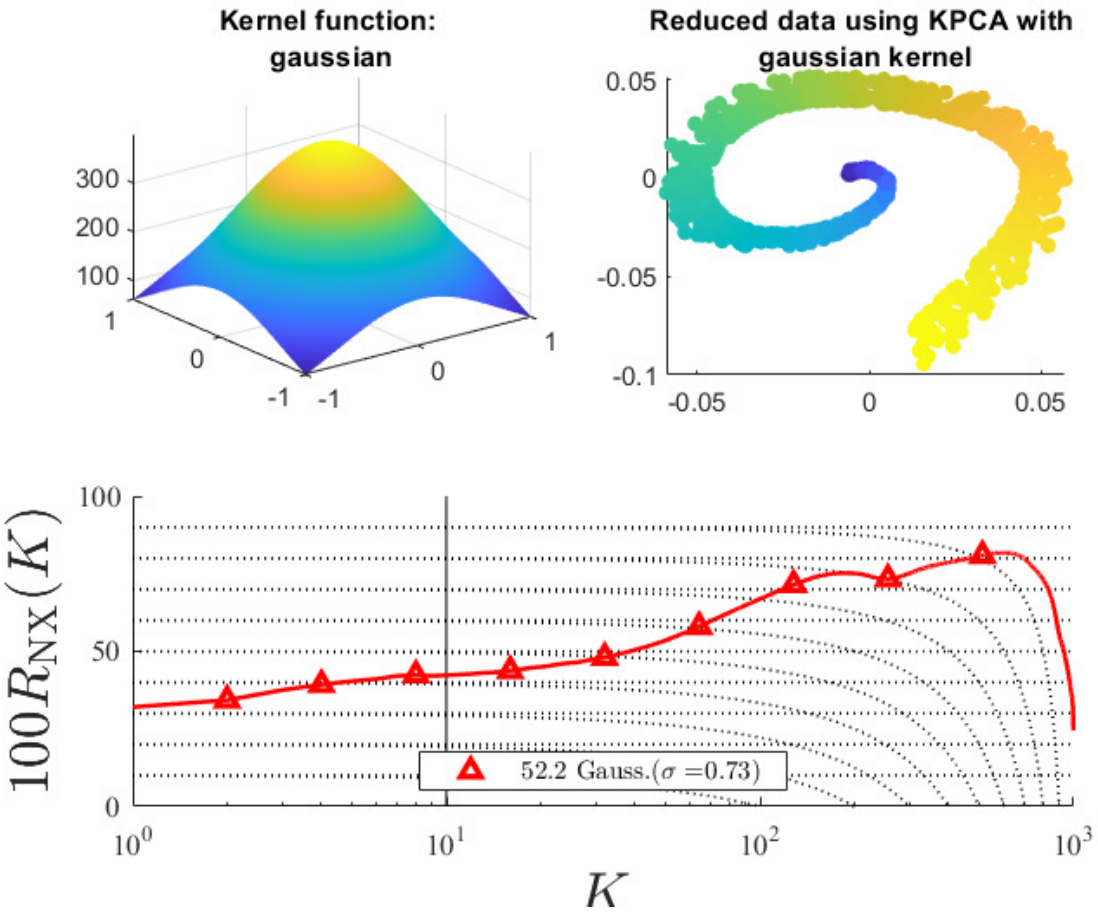

Figure 5.7: Structure Preservation Results for `SwissRoll` Dataset Trial 2

Same as in previous trial, kernel is able to capture only big structures Figure 5.7.

5.2.6 Class Separability Results

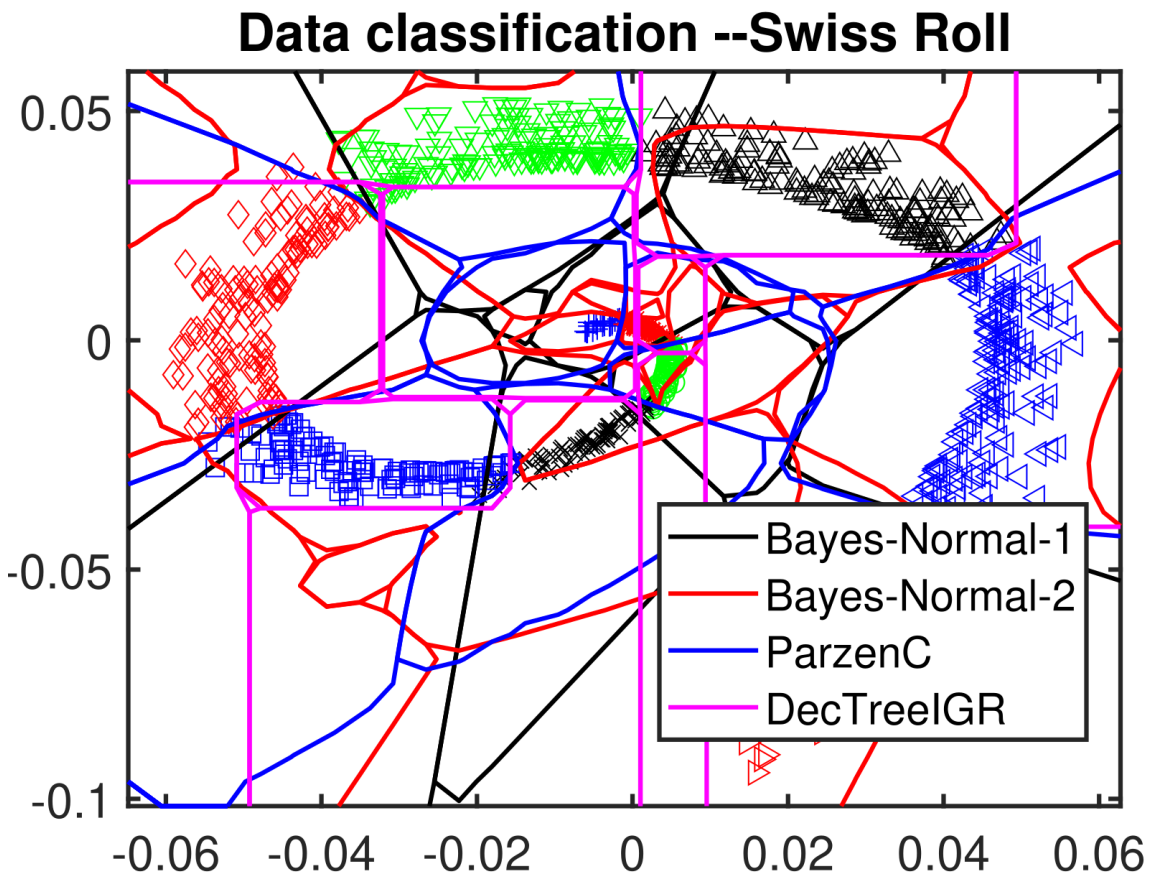

Figure 5.8: Class Separability Results for `SwissRoll` Dataset Trial 2

The best performant classifier is Bayes-Normal-2 with an accuracy of 96.69%. The embedded data shows separable classes.

5.2.7 Trial 3:

5.2.8 Structure Preservation Results

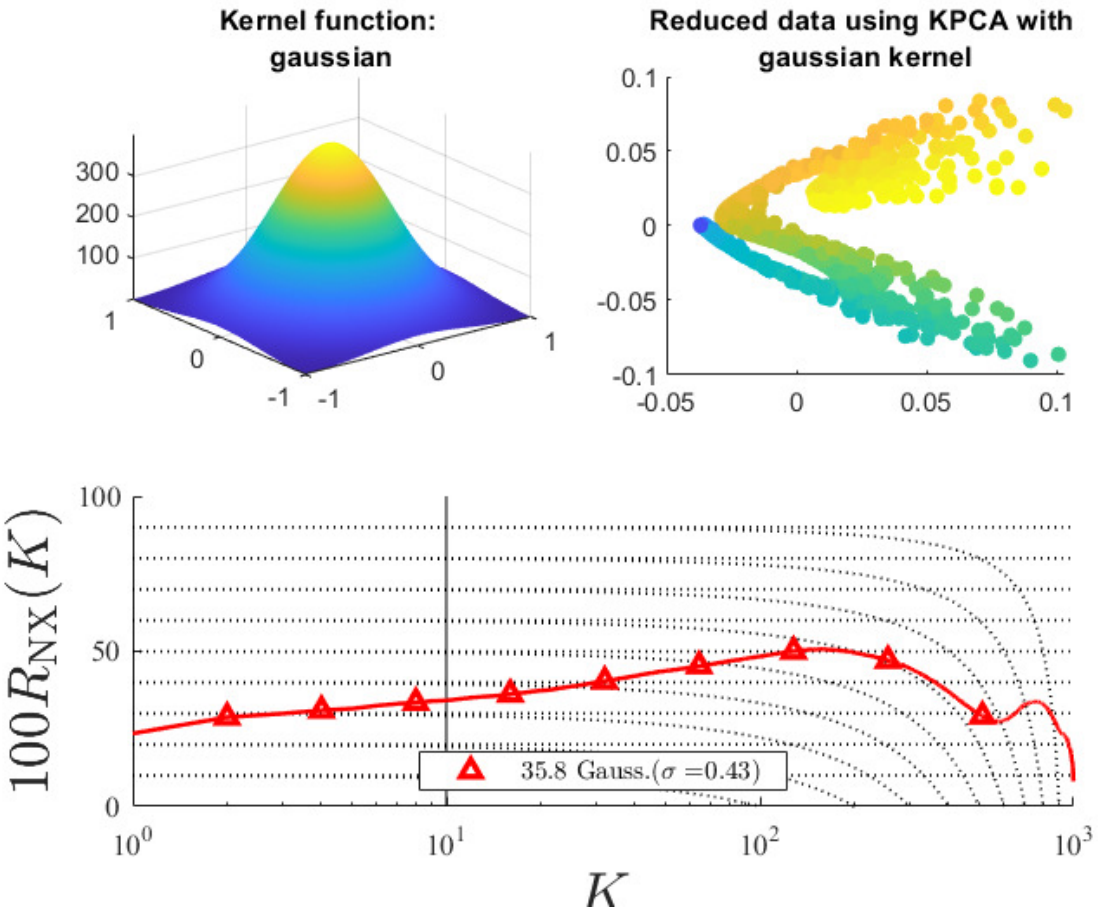

Figure 5.9: Structure Preservation Results for `SwissRoll` Dataset Trial 3

Poor embedding, as can be shown in the up-right image Figure **??** where data points overlapping is presented. Also in the Rnx graph for all values of k is not able to capture any structures (global or local)

5.2.9 Class Separability Results

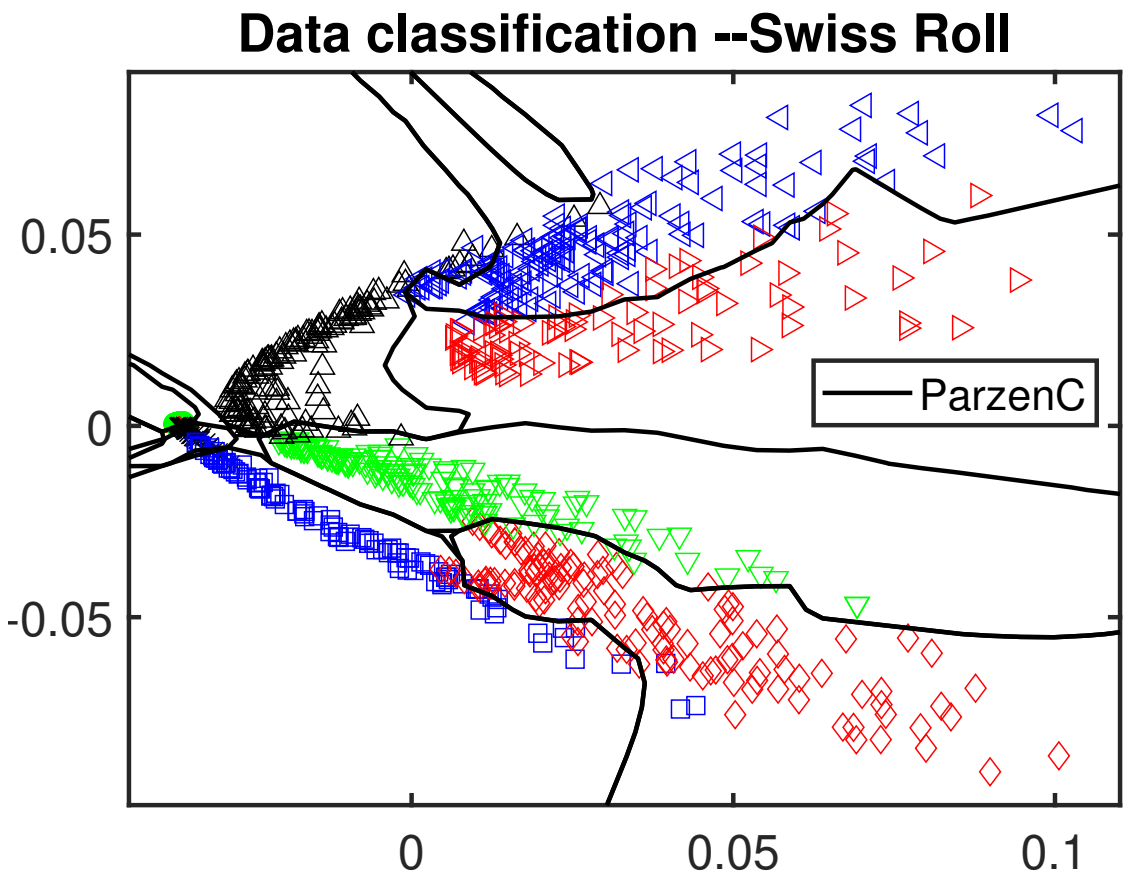

Figure 5.10: Class Separability Results for `SwissRoll` Dataset Trial 3

The best performant classifier is ParzenC in Figure 5.10 with an accuracy of 78.24%. Because of the overlapping the classifier is, somehow, complex.

5.2.10 Trial 4:

5.2.11 Structure Preservation Results

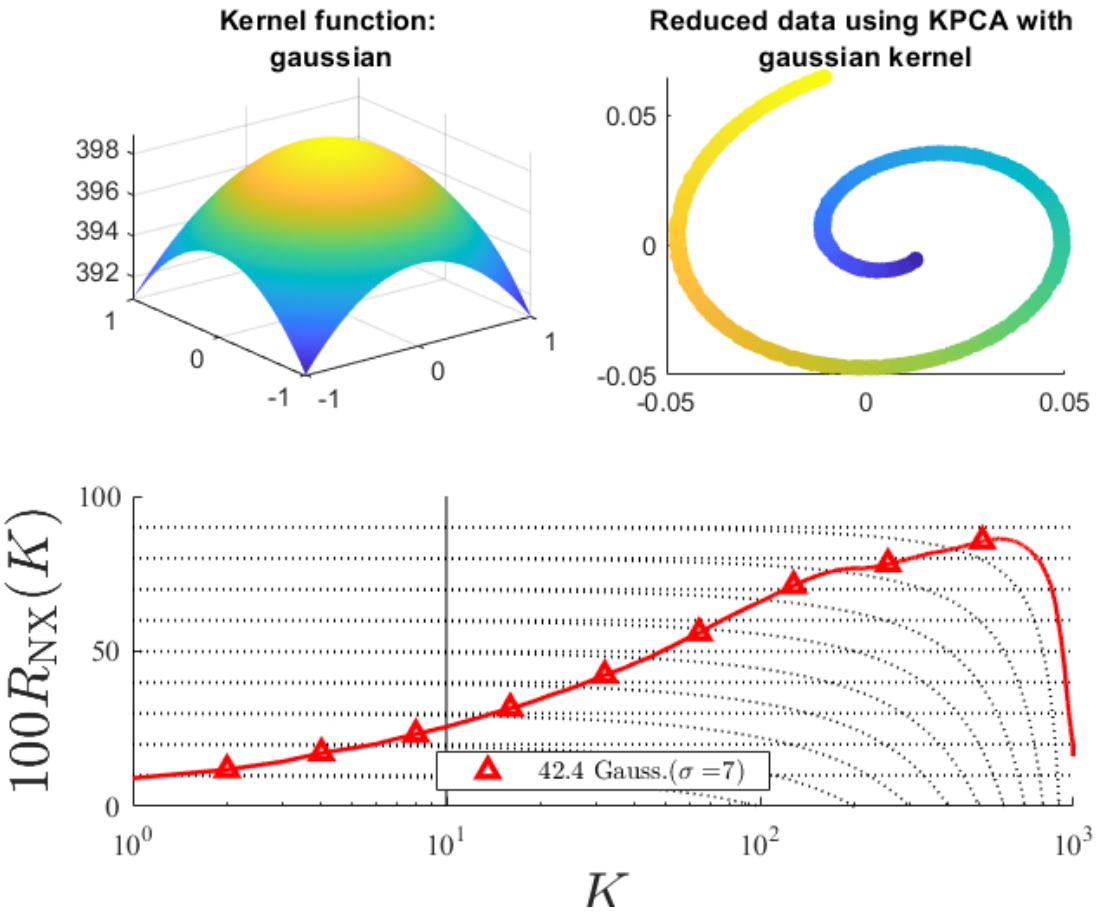

Figure 5.11: Structure Preservation Results for `SwissRoll` Dataset Trial 4

The Figure 5.11 shows a better embedding for bigger values of k, with nice separable classes.

5.2.12 Class Separability Results

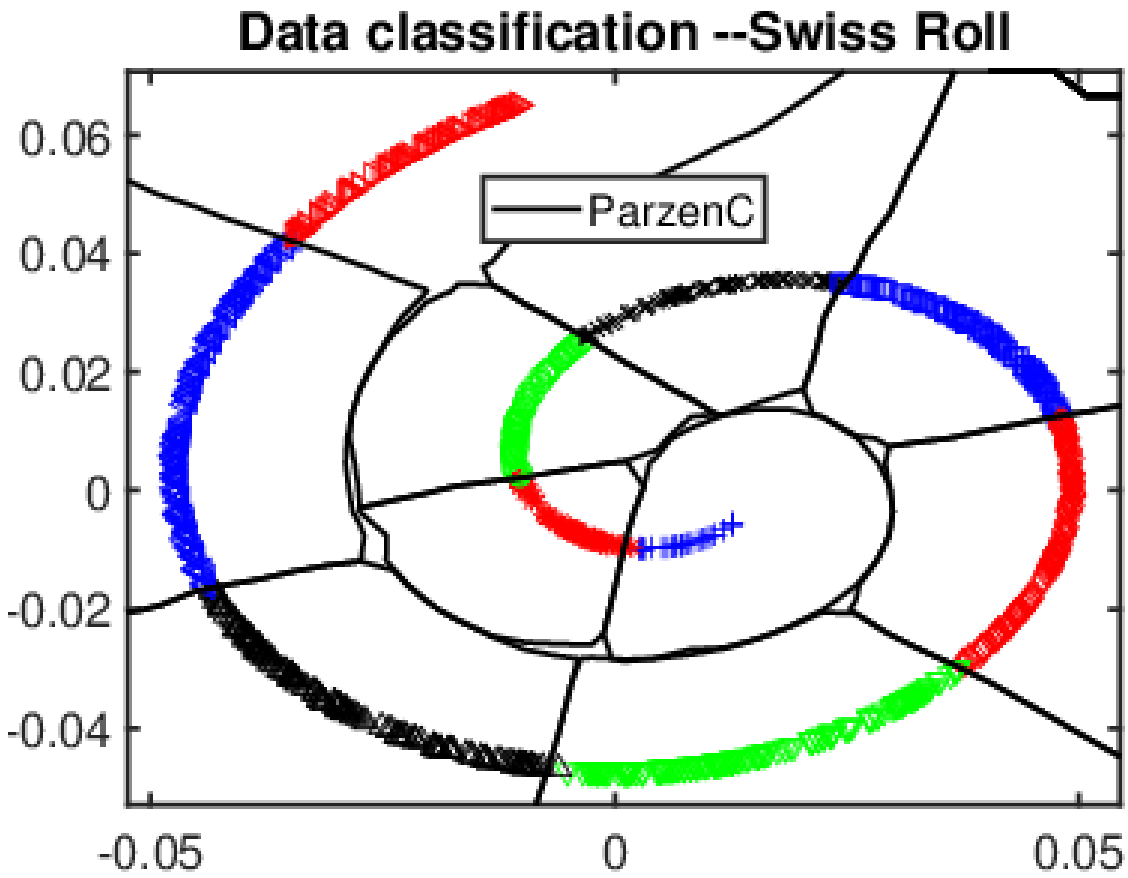

Figure 5.12: Class Separability Results for `Helix` Dataset Trial 4

The best performant classifier is ParzenC with an accuracy of 98.56%. Very powerful due to the (nice) separability of classes. As shown in Figure 5.12.

5.3 Experiment 3 : `Toroidal` Dataset

In this experiment we are using the `Toroid` dataset (not-normalized). The trial settings summary is as shown in Table 5.3.

Table 5.3: Trial settings for `Toroid` dataset

Trial	Kernel function	Parameters	Classifier Accuracy
1	Gaussian	$\sigma = 8.14$	ParzenC: 0.0060

5.3.1 Structure Preservation Results

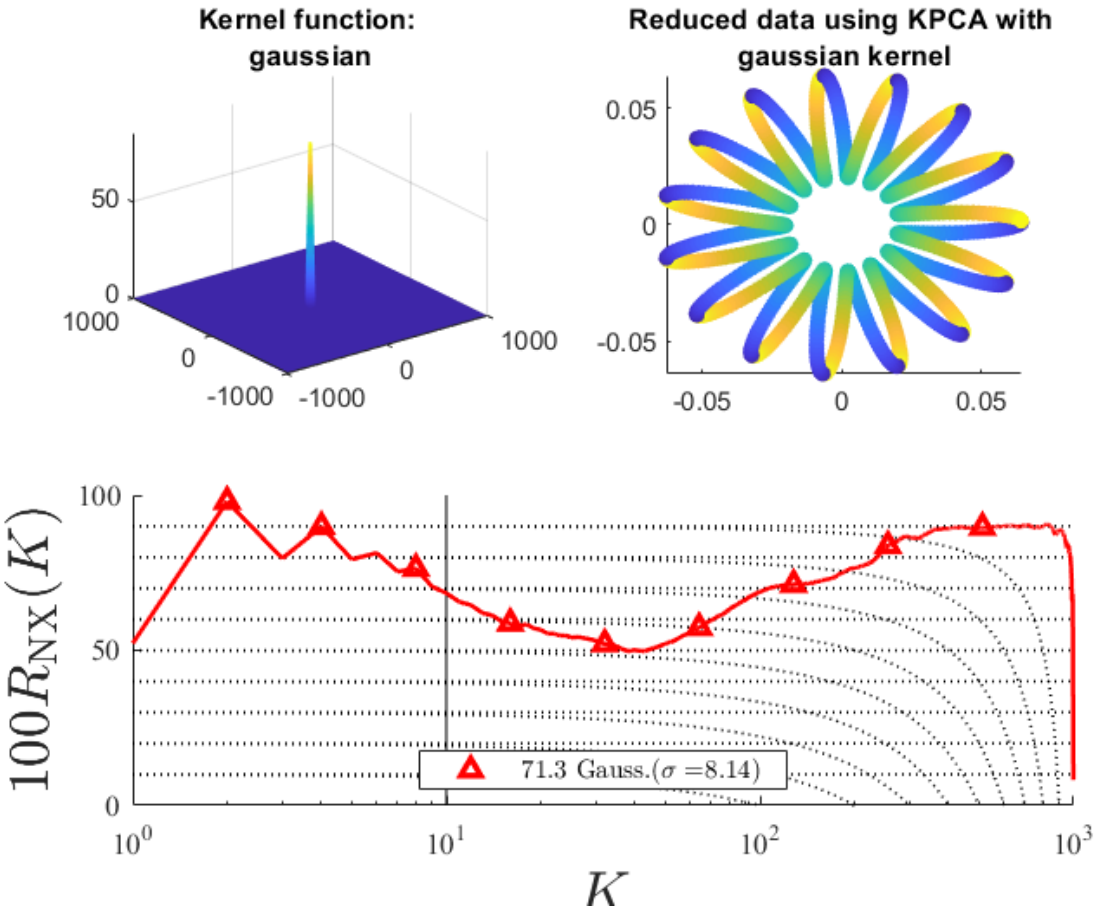

Figure 5.13: Structure Preservation Results for `Toroid` Dataset Trial 1

As it's clear from the up-right Figure 5.13 that the embedding was successful and it did capture all the important global and local structures. This is supported by the Rnx curve where for all values of k the AUC is quite interesting.

5.3.2 Class Separability Results

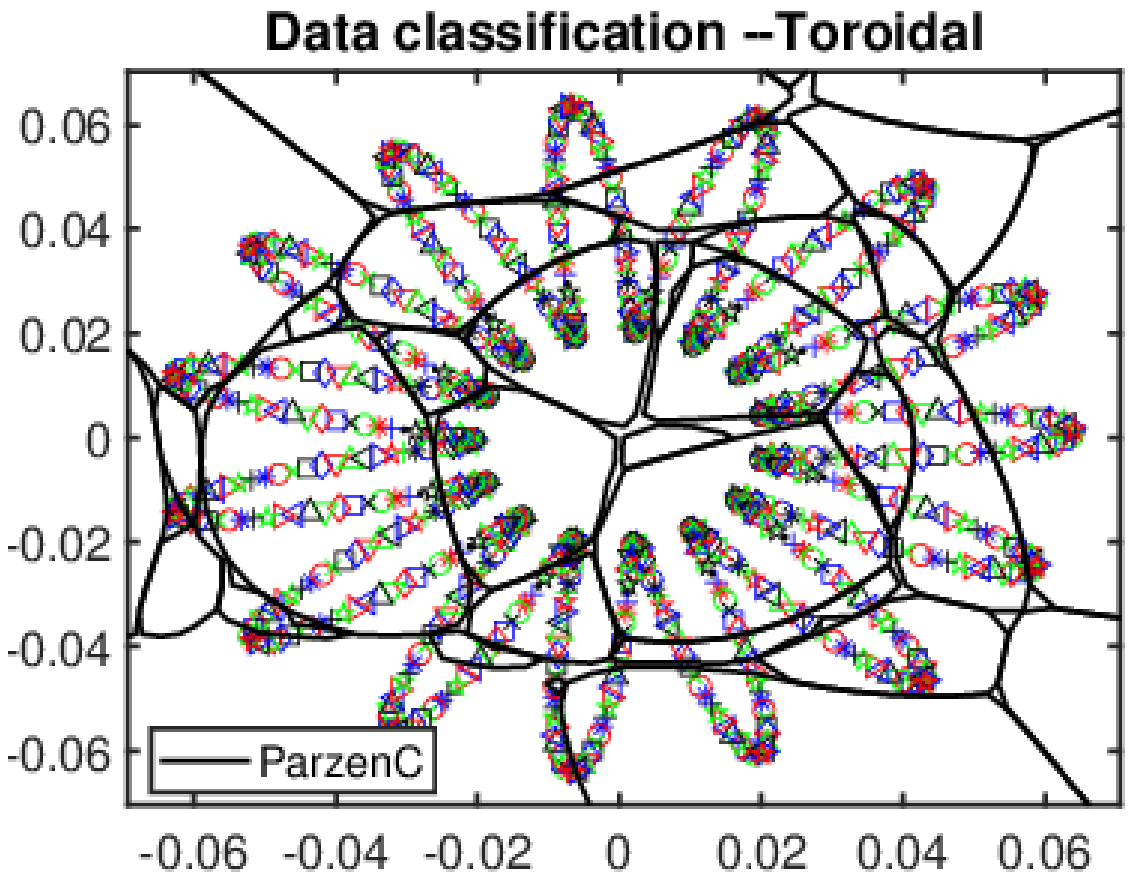

Figure 5.14: Class Separability Results for `Toroid` Dataset Trial 1

Even if the structure was very good, the kernel was not able to separate the classes accordingly. Therefore, the classification accuracy is not promising at all (00.60%).

Chapter 6

Conclusion and Future Work

As a conclusion, it is worth mentioning that although the kernel methods are powerful in many real-life scenarios. There is still a lack of interpretability of their result. In fact, there is a need for interactive applications that would enhance the understanding of data, especially for normal users or domain experts with no technical or theoretical background. Additionally, the theory behind the kernel is quite impressive and powerful. That's why is being used everywhere not only for PCA.

Finally, future work can be made over the presented work is to build interactive software that helps normal users play with their data in terms of applying different kernel functions, changing parameters, and so on, in order to make the data speak its hidden insights.

Bibliography

[1] S. Ahmad, P. Singh, and A. K. Sagar, "A survey on big data analytics," in *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, 2018, pp. 256–260.

[2] M. Zastrow, "Data visualization: Science on the map," *Nature*, vol. 519, no. 7541, pp. 119–120, 2015. [Online]. Available: https://doi.org/10.1038/519119a

[3] A. Anand, M. A. Haque, J. S. R. Alex, and N. Venkatesan, "Evaluation of machine learning and deep learning algorithms combined with dimentionality reduction techniques for classification of parkinson's disease," in *2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 2018, pp. 342–347.

[4] S. Ayesha, M. K. Hanif, and R. Talib, "Overview and comparative study of dimensionality reduction techniques for high dimensional data," *Information Fusion*, vol. 59, pp. 44–58, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S156625351930377X

[5] M. D. Ritchie, L. W. Hahn, N. Roodi, L. R. Bailey, W. D. Dupont, F. F. Parl, and J. H. Moore, "Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer," *The American Journal of Human Genetics*, vol. 69, no. 1, pp. 138–147, 2001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0002929707614530

[6] N. S. Ibrahim and D. A. Ramli, "I-vector extraction for speaker recognition based on dimensionality reduction," *Procedia Computer Science*, vol. 126, pp. 1534–1540, 2018, knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 22nd International Conference, KES-2018, Belgrade, Serbia. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918314042

[7] K. K. Pandey and D. shukla, "Challenges of big data to big data mining with their processing framework," in *2018 8th International Conference on Communication Systems and Network Technologies (CSNT)*, 2018, pp. 89–94.

[8] D. Peluffo, J. Lee, and M. Verleysen, "Generalized kernel framework for unsupervised spectral methods of dimensionality reduction," 12 2014.

[9] M. D. Srinath, P. Rajasekaran, and R. Viswanathan, *Introduction to statistical signal processing with applications*. Prentice-Hall, Inc., 1995.

[10] L. der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 2579-2605, p. 85, 2008.

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

[12] D. Peluffo, J. Lee, M. Verleysen, J. Rodríguez, and G. Castellanos-Dominguez, "Unsupervised relevance analysis for feature extraction and selection a distance-based approach for feature relevance," 01 2014.

[13] B. Schölkopf and A. Smola, *Smola, A.: Learning with Kernels - Support Vector Machines, Regularization, Optimization and Beyond. MIT Press, Cambridge, MA*, 01 2001, vol. 98.

[14] J. A. Lee and M. Verleysen, "Quality assessment of dimensionality reduction: Rank-based criteria," *Neurocomputing*, vol. 72, no. 7, pp. 1431–1443, 2009.

Appendices

Appendix A

Demonstration of the inner product equivalence

Showing that $\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{v} = \langle \boldsymbol{x}, \boldsymbol{v} \rangle \boldsymbol{x}^T$:

$$
\begin{aligned}
\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{v} &= \begin{pmatrix} (\boldsymbol{x}_1\boldsymbol{v}_1 + \boldsymbol{x}_2\boldsymbol{v}_2 + \ldots + \boldsymbol{x}_n\boldsymbol{v}_n)\,\boldsymbol{x}_1 \\ (\boldsymbol{x}_1\boldsymbol{v}_1 + \boldsymbol{x}_2\boldsymbol{v}_2 + \ldots + \boldsymbol{x}_n\boldsymbol{v}_n)\,\boldsymbol{x}_2 \\ \vdots \\ (\boldsymbol{x}_1\boldsymbol{v}_1 + \boldsymbol{x}_2\boldsymbol{v}_2 + \ldots + \boldsymbol{x}_n\boldsymbol{v}_n)\,\boldsymbol{x}_n \end{pmatrix} \\
&= (\boldsymbol{x}_1\boldsymbol{v}_1 + \boldsymbol{x}_2\boldsymbol{v}_2 + \ldots + \boldsymbol{x}_n\boldsymbol{v}_n) \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_n \end{pmatrix} \\
&= \langle \boldsymbol{x}, \boldsymbol{v} \rangle \boldsymbol{x}^T. \qquad \square
\end{aligned}
$$

Appendix B

Matlab Interface

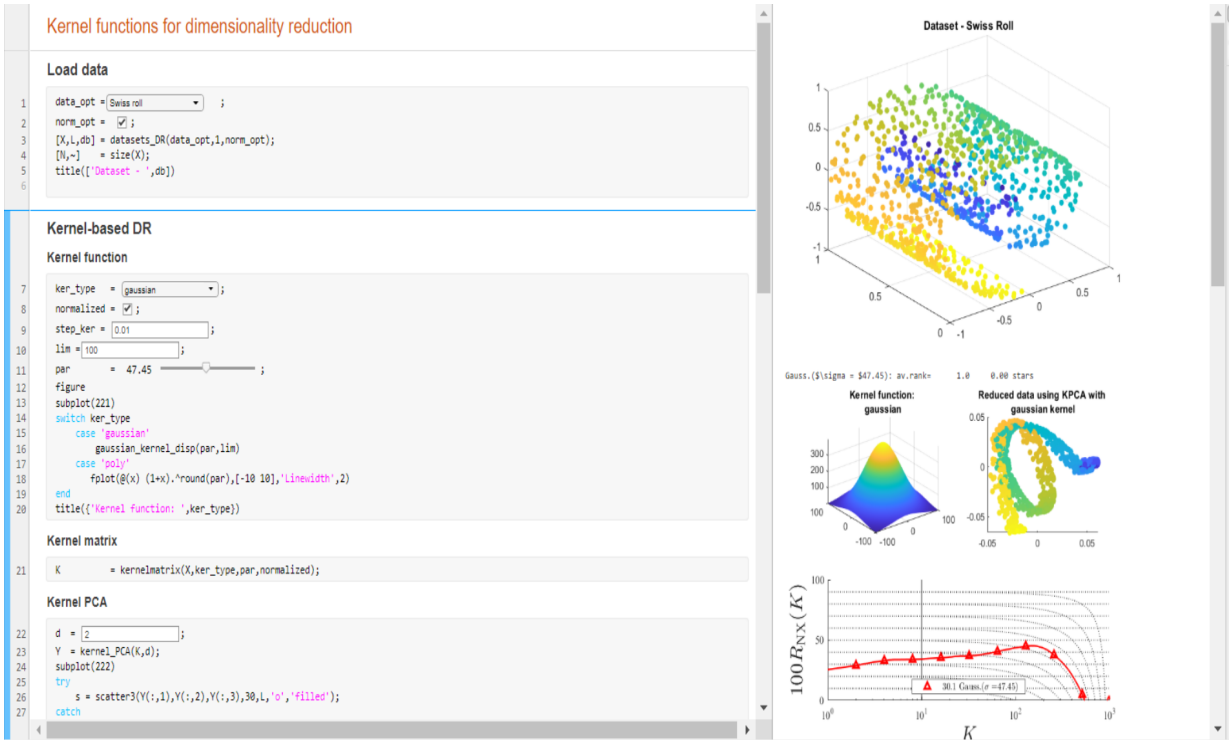

Figure B.1: Screenshot of the MATLAB live script created.