Advanced Machine Learning
Practical 1 Solution: Manifold Learning (PCA and Kernel PCA)

Professor: Aude Billard
Assistants: Bernardo Fichera, Roland Schwan
E-mails: aude.billard@epfl.ch,
bernardo.fichera@epfl.ch, roland.schwan@epfl.ch

Spring Semester 2021

1 Introduction

During this week's practical we will focus on understanding kernel Principal Component Analysis (Kernel PCA) by (i) comparing it to Principal Component Analysis (PCA), (ii) analyzing the role of the chosen kernel and hyper-parameters and (iii) applying it to high-dimensional overlapping real-world datasets.

2 ML_toolbox

`ML_toolbox` contains a set of methods and examples for easily learning and testing machine learning methods on your data in MATLAB. It is available in the following link:

`https://github.com/epfl-lasa/ML_toolbox`

From the course Moodle webpage (or the website), the student should download and extract the .zip file named `TP1(PCA,kPCA).zip` which contains the following files:

Code	Datasets
TP1_kPCA_2D.m	breast-cancer-wisconsin.csv
TP1_kPCA_HighD.m	ionosphere.csv
setup_TP1.m	hayes-roth.csv

Before proceeding make sure that `./ML_toolbox` is at the same directory level as the TP directory `./TP1-PCA+kPCA`. You must also place `setup_TP1.m` at the same level. Now, to add these directories to your search path, type the following in the MATLAB command window:

```
>> setup_TP1
```

NOTE: To test that all is working properly with ML_Toolbox you can try out some examples of the toolbox; look in the **examples** sub-directory.

3 Manifold Learning Techniques

Manifold Learning is a form of non-linear dimensionality reduction widely-used in machine learning applications to project data onto a nonlinear representation of smaller dimensions. Such techniques are commonly used as:

- Pre-processing step for clustering or classification algorithms to reduce the dimensionality and computational costs and improve performance.
- Feature extraction.

In *dimensionality reduction* problems, given a training dataset $\mathbf{X} \in \mathbb{R}^{N \times M}$ composed of M datapoints with N-dimensions, we would like to find a lower-dimensional embedding $\mathbf{Y} \in \mathbb{R}^{p \times M}$, through a mapping function $f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^N \rightarrow \mathbf{y} \in \mathbb{R}^p$ where $p < N$. This mapping function can be linear or non-linear, in this practical we will cover an instance of a linear approach (i.e. **PCA**) and its non-linear variant (i.e. **Kernel PCA**).

3.1 Principal Component Analysis (PCA)

In Principal Component Analysis (PCA) the mapping function $f(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}^{p \leq N}$ is given by the following linear projection

$$\mathbf{y} = f(\mathbf{x}) = A\mathbf{x}. \tag{1}$$

where $A \in \mathbb{R}^{p \times N}$ is the projection matrix, found by diagonalizing an estimate of the Covariance matrix $\mathbf{C}$ of the centered dataset $\sum_{i=1}^{M} \mathbf{x}_i = 0$:

$$\mathbf{C} = \frac{1}{M} \sum_{i=1}^{M} (\mathbf{x}^i)(\mathbf{x}^i)^T, \tag{2}$$

By extracting its eigenvalue decomposition $C = \mathbf{V}\Lambda\mathbf{V}^T$, the projection matrix A is constructed as $A = \mathbf{V}^T$ where $\mathbf{V} = [\mathbf{v}^1, \ldots, \mathbf{v}^N]$. To reduce dimensionality, one then chooses a sub-set of p eigenvectors $\mathbf{v}^i$ from $\mathbf{V}$. p can be determined by visualizing the projections or analyzing the Eigenvalues. The chosen Eigenvectors, i.e. the *Principal Components*, represent a new basis which maximize the variance along which the data is most spread.

For a thorough description of PCA, its application and derivation, refer to the PCA slides from the Applied Machine Learning Course.

3.2 Kernel Principal Component Analysis (Kernel PCA)

PCA assumes a linear transformation in the data and, as such, it can only work well if the data is linearly separable or linearly correlated. When dealing with non-linear inseparable data one requires a non-linear technique. To this end, [1] proposed a non-linear technique which exploits *kernel methods* to find a non-linear transformation that lifts that data to a high-dimensional *feature space* $\phi(\mathbf{x}) : \mathbb{R}^N \to \mathbb{R}^F$, where the linear operations of PCA can be performed. Kernel PCA begins with the Covariance matrix of the data-points projected to feature space and centered; i.e. $\sum_{i=1}^{M} \phi(\mathbf{x}^i) = 0$:

$$\mathbf{C} = \frac{1}{M} \sum_{i=1}^{M} \phi(\mathbf{x}^i)\phi(\mathbf{x}^i)^T, \tag{3}$$

As in PCA, one must then find eigenvalues $\lambda_k \geq 0$ and Eigenvectors $\mathbf{v} \in \mathbb{R}^F$, that satisfy $\lambda_k \mathbf{v}_k = C\mathbf{v}_k$ for all $k = 1, \ldots, M$. Given that $\mathbf{v}_k$ lies in the span of $\phi(\mathbf{x}^1), \ldots, \phi(\mathbf{x}^M)$ the problem becomes:

$$\lambda \langle \phi(\mathbf{x}^k), \mathbf{v}^k \rangle = \langle \phi(\mathbf{x}^k), C\mathbf{v}^k \rangle \quad \forall \quad k = 1, \ldots, M. \tag{4}$$

This implies that $\mathbf{v}^k = \sum_{i=1}^{M} \alpha_i^k \phi(\mathbf{x}^i)$. Via the kernel trick $k(\mathbf{x}, \mathbf{x}^k) = \langle \phi(\mathbf{x}), \phi(\mathbf{x}^k) \rangle$ and some substitutions (see [1]), the Eigendecomposition of (3) becomes the following dual Eigenvalue problem:

$$\lambda_k M \alpha^k = \mathbf{K}\alpha^k \tag{5}$$

where $\mathbf{K}_{ij} = k(\mathbf{x}^i, \mathbf{x}^j)$ is the kernel matrix, otherwise known as the Gram matrix and α is a column vector of M entries $\alpha^k = [\alpha_1^k, \ldots, \alpha_M^k]$. Diagonalizing, and hence solving for the Eigenvectors of $\tilde{\mathbf{K}} = \tilde{\mathbf{V}} \Lambda \tilde{\mathbf{V}}^T$ (after centering and normalizing $\mathbf{K} \to \tilde{\mathbf{K}}$), is equivalent to finding the coefficients α. We can then extract the desired p principal components, which, as in PCA, will correspond to the Eigenvectors with the largest eigenvalues. However, as opposed to PCA, extracting the principal components is not a straight-forward projections. This is due to the fact that, the Eigenvectors of $\mathbf{K}$ represent the data points already projected onto the respective principal components. Hence, one must compute the projection of the image of each point $\phi(\mathbf{x})$ onto each k-th $\mathbf{v}^k$ Eigenvector (for $k = 1, \ldots, p$) as follows:

$$\langle \mathbf{v}^k, \phi(\mathbf{x}) \rangle = \sum_{i=1}^{M} \alpha_i^k \langle \phi(\mathbf{x}), \phi(\mathbf{x}^i) \rangle = \sum_{i=1}^{M} \alpha_i^k k(\mathbf{x}, \mathbf{x}^i). \tag{6}$$

Thus, the p-dimensional projection of a point $\mathbf{x}$ is the vector $\mathbf{y} = [y_1, \ldots, y_p]$ for $y_k \in \mathbb{R}$, where each y_k is computed as follows:

$$y_k(\mathbf{x}) = \langle \mathbf{v}^k, \phi(\mathbf{x}) \rangle = \sum_{i=1}^{M} \alpha_i^k k(\mathbf{x}, \mathbf{x}^i) \quad \forall \quad k = 1, \ldots, p \tag{7}$$

Since the Eigenvectors are computed from the Gram matrix $\mathbf{K} \in \mathbb{R}^{M \times M}$, the number of principal components to keep can be $p \leq M$; as opposed to PCA where $p \leq N$ due to $\mathbf{C} \in \mathbb{R}^{N \times N}$. Until now, we have not defined $k(\mathbf{x}, \mathbf{x}^i)$, Kernel PCA has the flexibility of handling many types of kernels, as long as it complies with Mercer's Theorem [2].

Kernels We have three options implemented in **ML_toolbox**:

- **Homogeneous Polynomial:** $k(\mathbf{x}, \mathbf{x}^i) = (\langle \mathbf{x}, \mathbf{x} \rangle)^d$,
  where $d < 0$ and corresponds to the polynomial degree.
- **Inhomogeneous Polynomial:** $k(\mathbf{x}, \mathbf{x}^i) = (\langle \mathbf{x}, \mathbf{x} + c \rangle)^d$,
  where $d < 0$ and corresponds to the polynomial degree and $c \geq 0$, generally $c = 1$.
- **Radial Basis Function (Gaussian):** $k(\mathbf{x}, \mathbf{x}^i) = \exp\left\{-\frac{1}{2\sigma^2}||\mathbf{x} - \mathbf{x}^i||^2\right\}$,
  where σ is the width or scale of the Gaussian kernel centered at $\mathbf{x}^i$

4 Kernel PCA Analysis

4.1 Projecting Datasets with PCA & Kernel PCA in ML_Toolbox

In the MATLAB script `TP1_kPCA_2D.m` we provide an example of loading datasets and applying PCA and Kernel PCA to generate lower-dimensional embeddings of the loaded datasets. By running the **first** code sub-block `1(a)`, the 3D Random Clusters Dataset shown in Figure 1 will be loaded to the current MATLAB workspace.

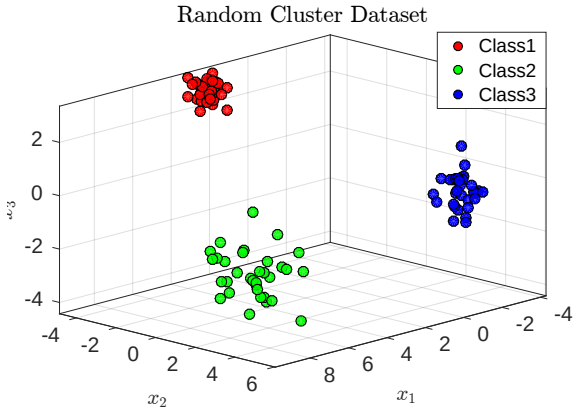

Figure 1: 3D Random Clusters Dataset.

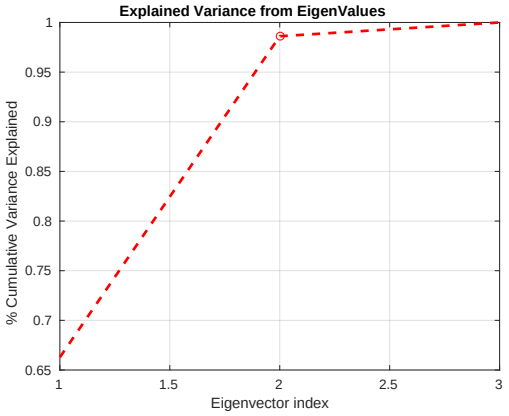

Figure 2: Explained Variance plot of 3D Random Clusters Dataset; $p = 2$ for $\sigma > 90\%$.

Principal Component Analysis (PCA): By running the **second** code sub-block `2(a)`, the Eigenvectors $\mathbf{V}$ and Eigenvalues Λ of the Covariance matrix $\mathbf{C}$ pertaining to the PCA algorithm are computed. This block will also plot the explained variance shown in Figure 2 through the `ml_explained_variance.m` function. One can also visualize the eigenvalues with the `ml_plot_eigenvalues.m` function. In code sub-block `2(b)`, p is chosen to be 3, the projected data-points $\mathbf{y} = A\mathbf{x}$ are computed and visualized in a scatter plot as in Figure 3. The diagonal plot correspond to the projections on a single Eigenvector $\mathbf{v}^i$ represented by histograms, whereas the off-diagonal plots correspond to the pair-wise combination of projections. The scatter matrix visualization (Figure 3)

can help us determine the necessary p to generate a linearly separable embedding. A typical approach to determine p is to select the number of Eigenvectors that can explain more than 90% of the variance of the data. As seen in Figure 2, this constraint yields $p = 2$. However, by analyzing the scatter matrix (Figure 3), one can see that the first Eigenvector can already describe a linearly separable distribution of the classes in the dataset.

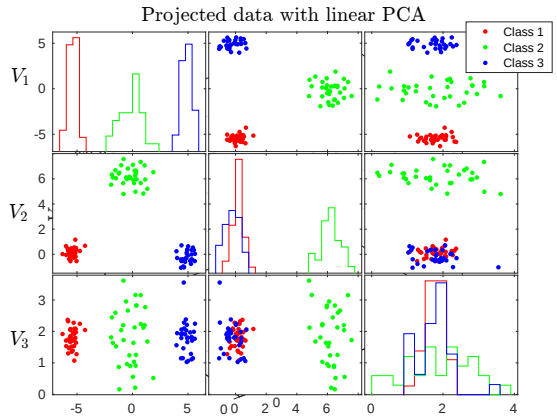

Figure 3: Principal Component Analysis (PCA) of 3D Random Clusters Dataset with $p = 3$.

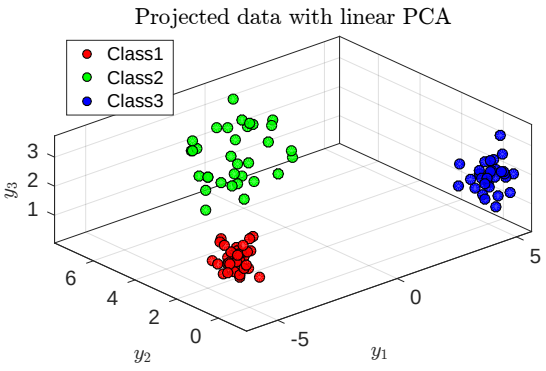

Figure 4: Projected Data of 3D Random Clusters Dataset PCA with $p = 3$.

When $p \leq 3$, the projected points can be visualized in Cartesian coordinates as in Figure 4. This plot can be obtained by modifying the plotting option defined in line 104, as:

```
1 plot_options.is_eig      = false;
```

Kernel Principal Component Analysis (Kernel PCA): In the **third** code block, one can find the necessary functions to extract the kernel principal components of a dataset. By running the **third** code sub-block `3(a)`, the kernel matrix $\mathbf{K}$ and its corresponding Eigenvectors α and Eigenvalues Λ are computed. One must define the following parameters: (i) the number of Eigenvectors to compute, theoretically this can be M, however this is computationally inefficient, hence, we choose a moderate value, i.e. between 10 and 20; (ii) the kernel type and (iii) kernel parameters, as follows:

```
% Compute kPCA with ML_toolbox
options = [];
options.method_name    = 'KPCA';    % Choosing kernel-PCA method
options.nbDimensions   = 10;        % Number of Eigenvectors to keep.
options.kernel         = 'gauss';   % Type of Kernel: {'poly', 'gauss'}
options.kpar           = [0.75];    % Variance for the RBF Kernel
                                    % For 'poly' kpar = [offset degree]
[kpca_X, mappingkPCA]  = ml_projection(X',options);
```

Once α and Λ are computed, one can visualize the Eigenvalues to determine an appropriate p with the `ml_plot_eigenvalues.m` as in Figure 5. For the given dataset, a value of $p = 3$ seems to be appropriate for the chosen kernel (RBF) and hyper-parameter

($\sigma = 0.75$), as the Eigenvalues λ_i stop decreasing drastically after λ_3. To estimate the projections of $\mathbf{X}$ with $p = 3$ one must then use (7) to compute the projection on each dimension. This is implemented in code sub-block `3(b)`, accompanied with the necessary functions to visualize the projections, as in Figure 6.

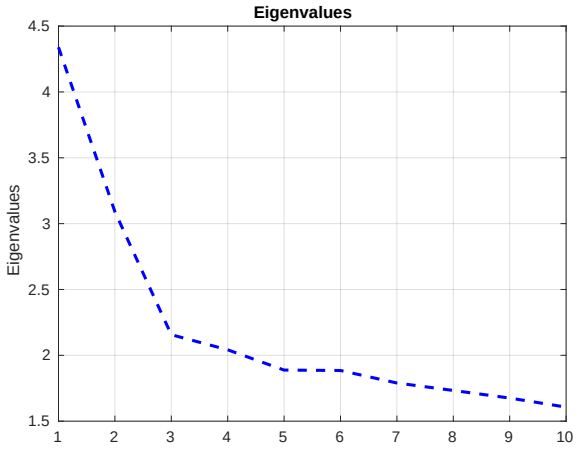

Figure 5: Eigenvalues of kPCA from 3D Random Clusters Dataset.

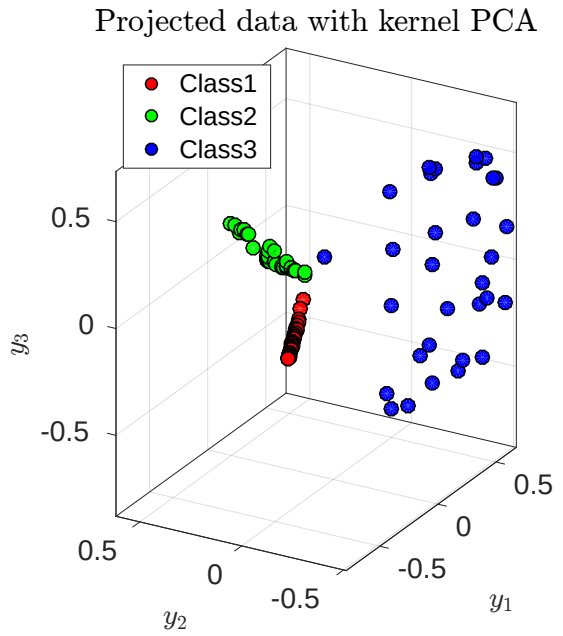

Figure 6: Projected Data of 3D Random Clusters Dataset kPCA with $p = 3$.

Dual Eigenvector Isoline Projection: Due to the special properties of kernel PCA, for each dual eigenvector $\mathbf{v}^k$, we can display the isolines (i.e. level curves) through (6). To recall, the isolines correspond to the region of the space for which the points have the same projection on the dual eigenvector. Moreover, we can superpose the data-points in original space on each Eigenvector projection. By running code sub-block `3(c)` one can generate such projections as the ones depicted in Figure 7. To choose the Eigenvectors to plot, one must modify the following isoline plotting option:

```
1 iso_plot_options.eigen_idx          = [1:6];  % Eigenvectors to use.
```

As can be seen, not unlike PCA, already with the first Eigenvector, the clusters are well-separated. Moreover, with the second Eigenvector each cluster belongs to a discriminative peak in the isolines; i.e. automatically achieving a form of feature extraction. To better visualize such phenomenon, instead of plotting the isolines with contours we can plot them with 3D surface plots, as shown in Figure 8 for the first two Eigenvectors. Such plots can be achieved by simply setting the following isoline plotting option to true:

```
1 iso_plot_options.b_plot_surf    = true; % Plot isolines as (3d) surface
```

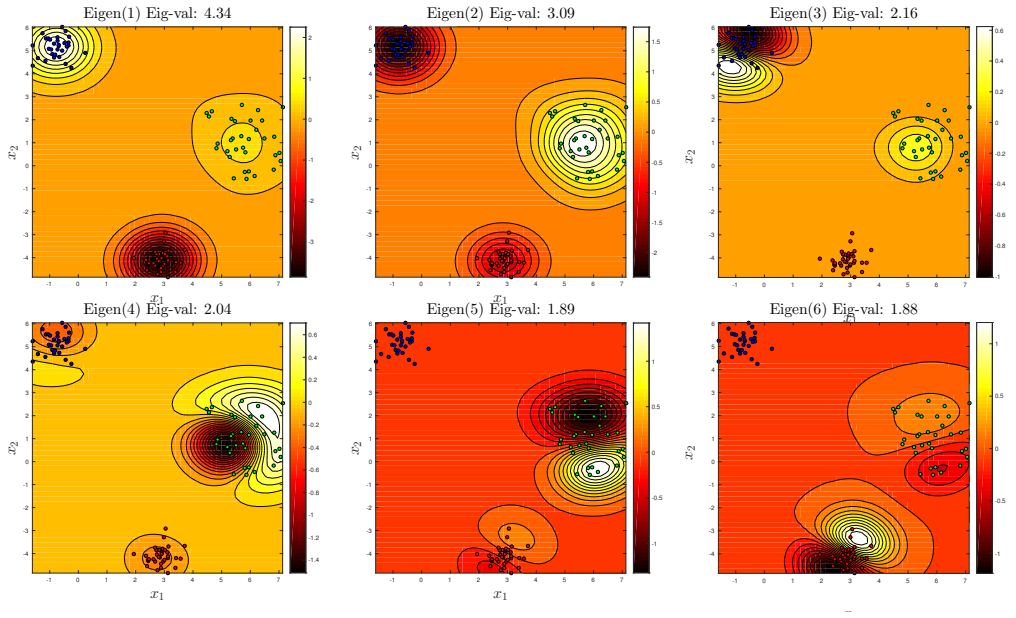

Figure 7: Isolines of the first $p = 6$ Eigenvectors of kPCA from the 3D Random Clusters Dataset (showing only the first 2 dimensions of the original data).

Grid Search of Kernel Parameters: In order to achieve a "good" embedding, one of the most important things to take into account is the kernel hyper-parameters. Given a dataset of 2/3D dimensions, it might be simple to find an appropriate value of, let's say the σ for the RBF Kernel by visually estimating how the data is spread and subsequently visualizing the dual Eigenvector isolines. This process, however, can be cumbersome and sometimes infeasible for high-dimensional data. To alleviate this, one can do a grid search on a range of parameters and determine the best value by analyzing the behavior of their corresponding eigenvalues, as in Figure 9. From this grid search, one can draw many conclusions. First of all, one can see that very large and very low values of σ have no impact on the outcome of the Eigenvalues, hence are not optimal. The behavior that we seek, is that of $\sigma = [1, 4.64]$, where the first Eigenvalues are very high and suddenly a drastic drop occurs, not surprisingly around λ_3. As will be seen in the next practical, this procedure is a good starting point to find the optimal clusters K, when unknown.

To run this grid search on your dataset, we have provided example code in sub-block `3(d)` of the MATLAB script: `TP1_kPCA_2D.m`. One can modify the range of parameters and type of kernel as follows:

```
grid_options = [];
grid_options.method_name        = 'KPCA';
grid_options.nbDimensions       = 10;
grid_options.kernel             = 'gauss';
% grid_options.kpar             = 0.4;      % if poly this is the offset

% Set Range of Hyper-Parameters
kpars = [0.001,0.01, 0.05,0.1,0.2, 0.5, 1, 2];
[ eigenvalues ] = ml_kernel_grid_search(X',grid_options,kpars);
```

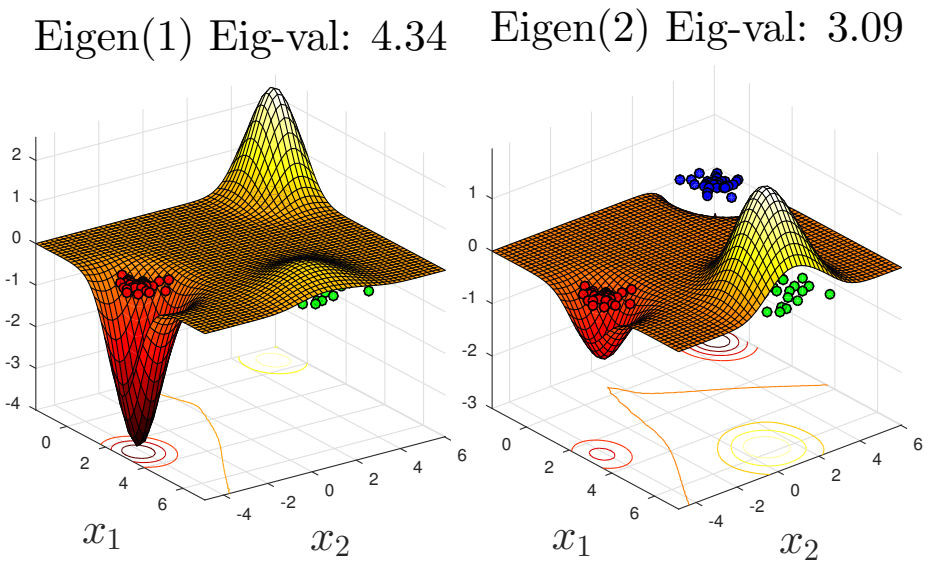

Figure 8: Isolines (Surface plot) of the first $p = 2$ Eigenvectors of kPCA from the 3D Random Clusters Dataset.

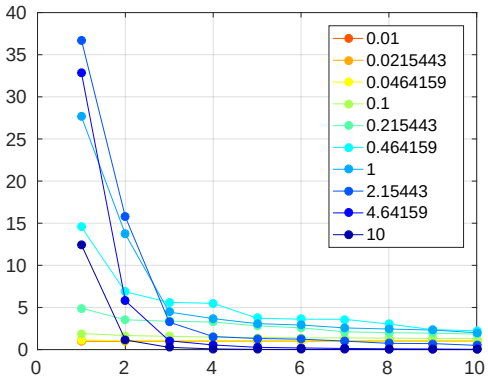

Figure 9: Grid Search of σ for the RBF Kernel.

4.2 Analysis of Kernel Choice and Hyper-parameters

TASK 1: Try Kernel PCA on Non-Linearly Separable 2D Toy Datasets
You must try to determine with which type of kernel and with which parameters you could generate a non-linear projection that can transform the dataset into an easily separable embedding, where a simple clustering/classification algorithm can be applied. The datasets in question are depicted in Figure 10 and 11. In order to address this, you should ask yourself the following questions:

- What kind of projection can be achieved with an RBF kernel and with a polynomial kernel?
- How should we relate the kernel width (σ) to the data available?
- What is the influence of the degree (d) of a polynomial kernel? Does it matter if the degree is even or odd?

Once you have answered these questions, load the datasets by running sub-blocks `1(b)` and `1(c)` of the accompanying MATLAB script: `TP1_kPCA_2D.m` and find a good projection where the classes are easily separable by modifying sub-blocks `3(a-c)` with your chosen kernel and parameters.

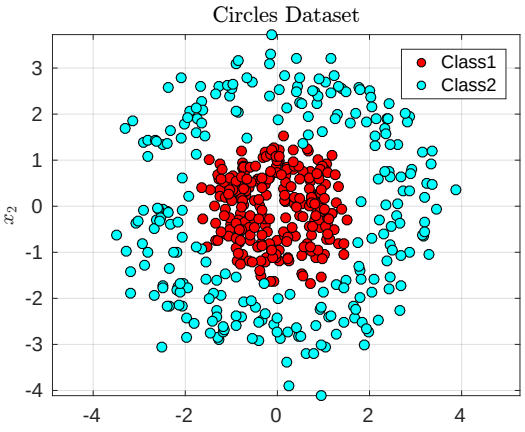

Figure 10: 2D Concentric Circles Dataset.

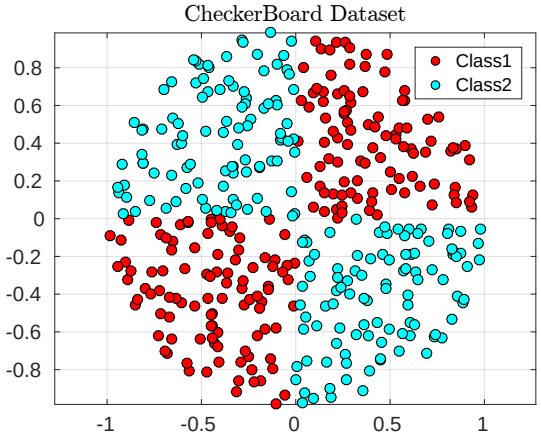

Figure 11: 2D CheckerBoard Dataset.

Once you have found good projections for these examples, try to create different shapes to get an intuition on which type of structure can be encapsulated by each kernel. You can draw 2D Data with the `ml_generate_mouse_data.m` function from ML_toolbox. We provide an example script to load the GUI in:

`ML_toolbox/functions/data_generation/ml_draw_data.m`.

By running this script you can load the drawing GUI shown in Figure 12. After drawing your data, you should click on the `Store Data` button, this will store a data array in your MATLAB workspace. After running the following sub-blocks the data and labels will be stored in two different arrays: $\mathbf{X} \in \mathbb{R}^{2\times M}$ and $\mathbf{y} \in \mathbb{I}^{M}$, which can then be used to visualize and manipulate in MATLAB (Figure 13).

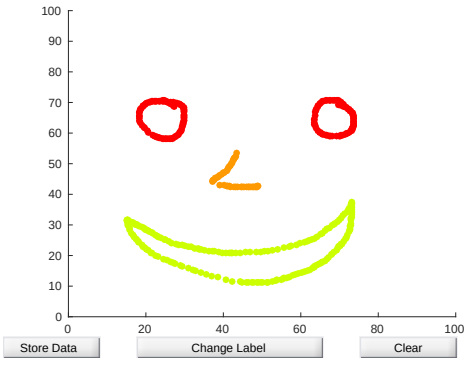

Figure 12: ML_toolbox 2D Data Drawing GUI.

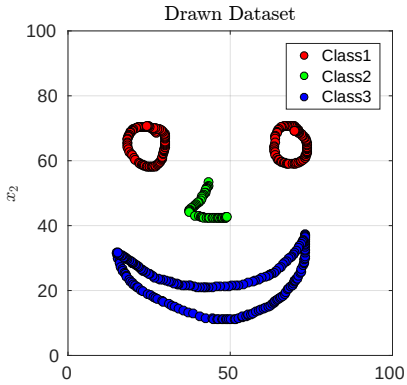

Figure 13: Data After Storing it in MATLAB workspace.

Try to create examples of data separable by both polynomial kernel and RBF kernel.

Solution Task 1:

For the Circles Dataset, the radial distribution of the data calls for an RBF kernel. Looking at the distribution of the data one would guess that a width around the variance of our data: $\sigma = [2, 4]$; i.e. inner and outer radius of the outer circle. Figure 14 and 15 show the resulting embedding from KPCA using $\sigma = 2$ and $\sigma = 4$, respectively. As can be seen, with both parameters KPCA can transform the data into a linearly separable dataset. A simple classifier such as a decision stump on y_3 would suffice to classify this data correctly.

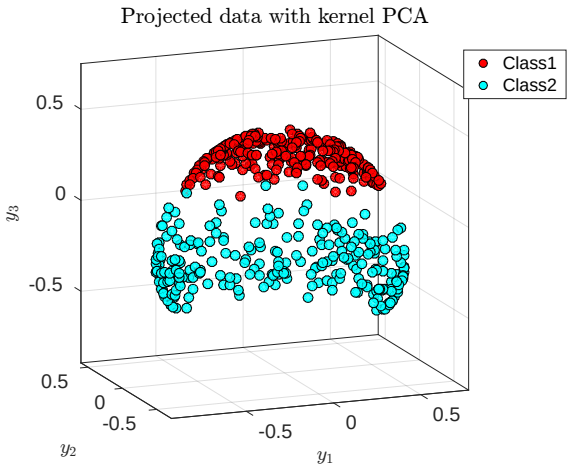

Figure 14: Projected Circles Dataset with KPCA, RBf Kernel $\sigma = 2$.

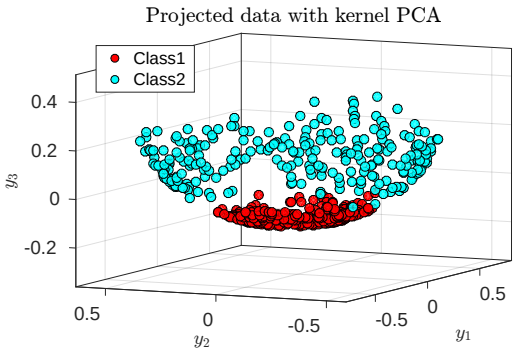

Figure 15: Projected Circles Dataset with KPCA, RBf Kernel $\sigma = 4$.

By plotting the isolines of the dual Eigenvectors as in Figure 16 and 17, we can see that the first and second Eigenvectors do a diagonal clustering of the datasets, while the third Eigenvector perfectly models the radial distribution of the dataset.

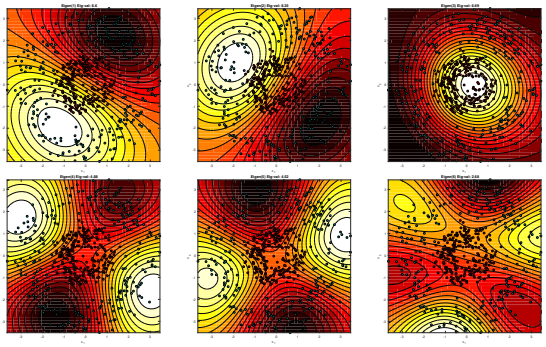

Figure 16: Isolines of first 6 Dual Eigenvectors for the Circles Dataset with KPCA, RBf Kernel $\sigma = 2$.

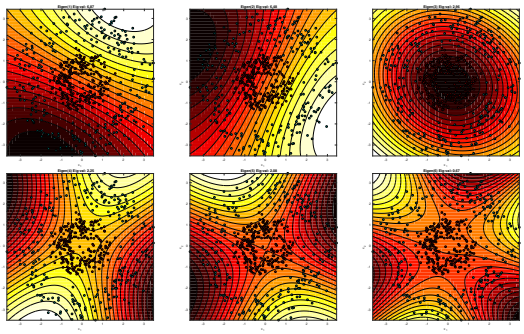

Figure 17: Isolines of first 6 Dual Eigenvectors for the Circles Dataset with KPCA, RBf Kernel $\sigma = 4$.

For the CheckerBoard Dataset, it seems that the data can be modeled by polynomials. The symmetry indicates that the degree should be of even order. With a polynomial

kernel of degree $d = 2$, on can achieve the projection depicted in Figure 18. As in the previous case, this projection generates a linearly separable dataset, with the decision boundary being a diagonal along y_1 and y_2. Taking a closer look at the isolines (Figure 20) we can see how the first and second projections of the Eigenvectors perfectly cluster the data on each side of the polynomial, this can be better seen of we plot the isolines as surface plots (shown in Figure 21). To validate our reasoning for choosing an even number polynomial degree, we show the projected dataset with a polynomial kernel of degree $d = 3$ in Figure 19. As can be seen, a third degree polynomial, does not necessarily exploit the symmetry in the original space.

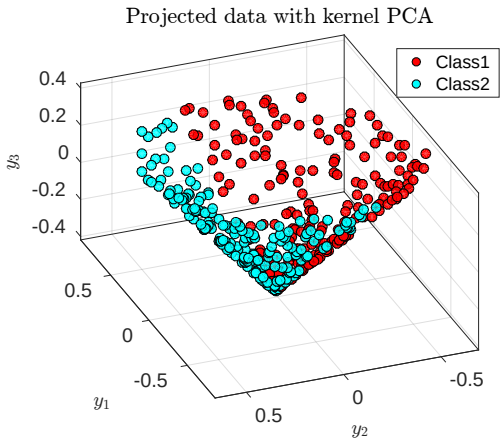

Figure 18: Projected Checkerboard Dataset with KPCA, Homogeneous Polynomial Kernel with $d = 2$.

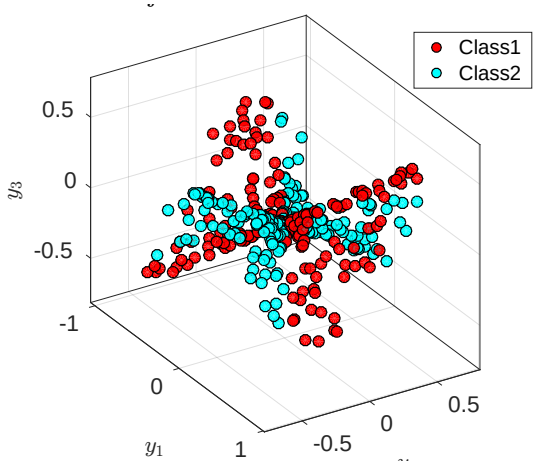

Figure 19: Projected Checkerboard Dataset with KPCA, Homogeneous Polynomial Kernel with $d = 3$.

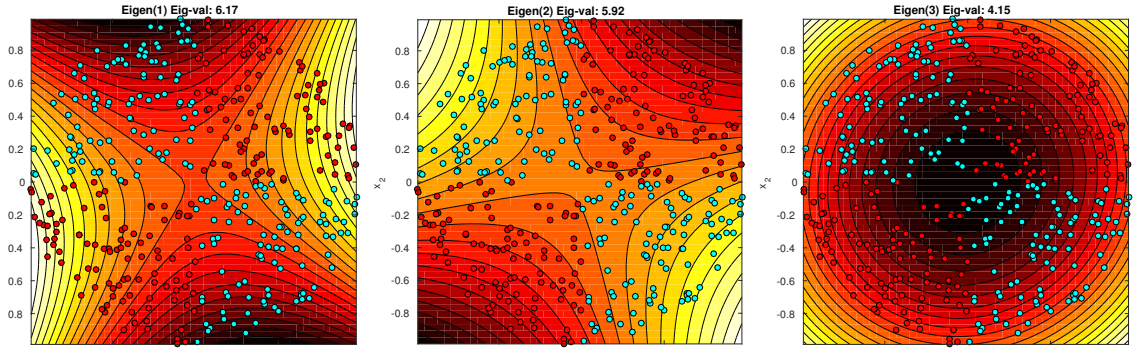

Figure 20: Isolines of first 6 Dual Eigenvectors for the Checkerboard Dataset with KPCA, Homogeneous Polynomial Kernel with $d = 2$.

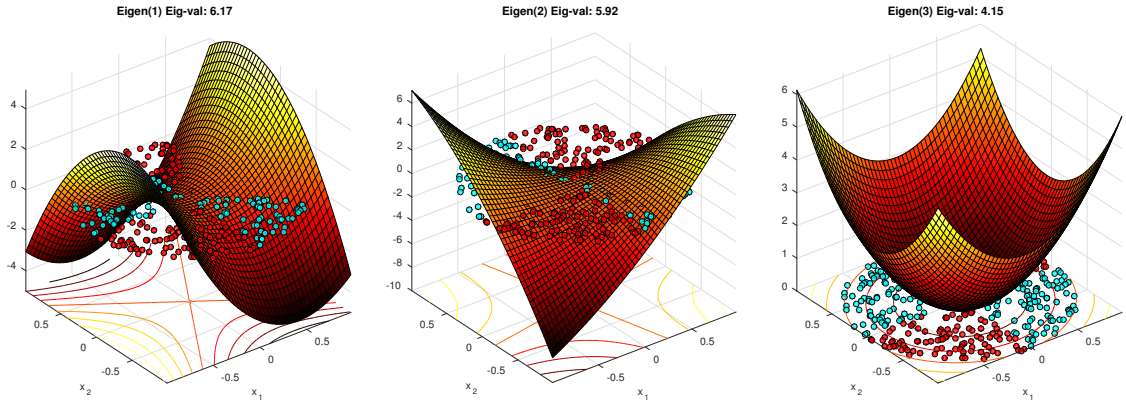

Figure 21: Isolines (Surface Plot) of first 6 Dual Eigenvectors for the Checkerboard Dataset with KPCA, Homogeneous Polynomial Kernel with $d = 2$.

For the drawing task on can draw datasets such as the ones in Figure 22 and 23.

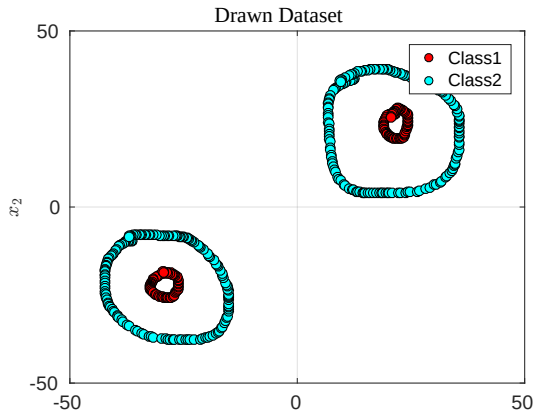

Figure 22: Example of Two Circles.

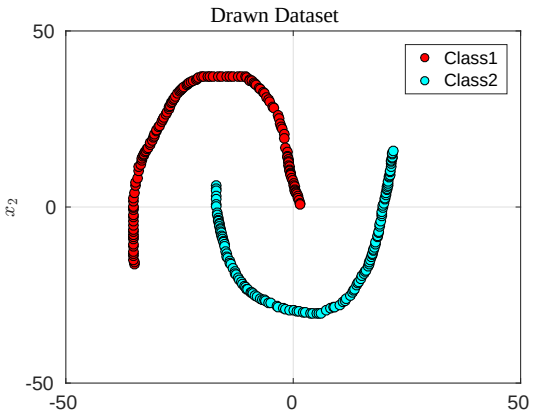

Figure 23: Example of Loops.

If we try the RBF Kernel on the two Circles dataset (Figure 22) with $\sigma = 50$ we can see how the original data can be transformed to different shapes (Figure 24). Notably, the Eigenvector combinations $\mathbf{v}^2 \times \mathbf{v}^3$ or $\mathbf{v}^3 \times \mathbf{v}^4$ we cluster seem to project each class to a common reference frame. Taking a close look at the isolines of the third Eigenvector, one can see that concentric circles lie on the same level, which in combination with the second or fourth Eigenvector result in a dataset similar to the concentric circles dataset in the previous example. Such transformation seems to be exploiting the symmetry of the dataset. When symmetries are involved, the simple polynomial kernel can achieve similar or (even better) transformations as the RBF kernel and with less computational burden. In Figure 26 we show the result of applying a second order polynomial kernel on the same dataset. As can be seen, the Eigenvector combination $\mathbf{v}^1 \times \mathbf{v}^2$ achieve a similar transformation as the RBF Kernel, with less computational effort.
For the loops dataset (Figure 23) we can try an RBF Kernel with $\sigma = 20$. As seen in

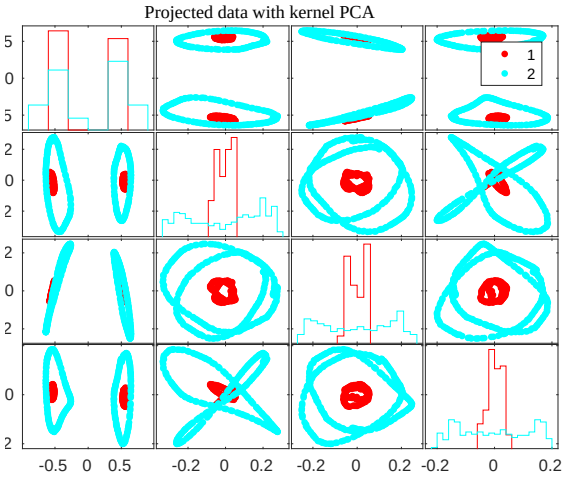

Figure 24: KPCA (RBF) projections for 2 Circles Dataset with $\sigma = 50$.

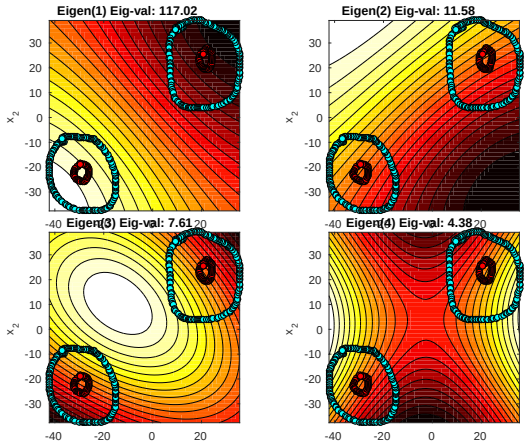

Figure 25: KPCA (RBF) Eigenvector Isolines for 2 Circles Dataset with $\sigma = 50$

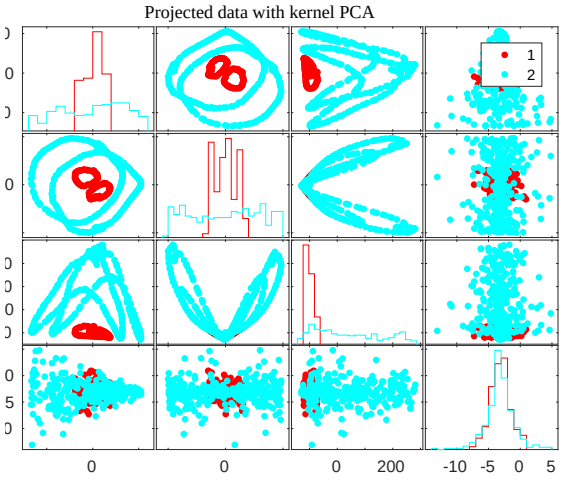

Figure 26: KPCA (Poly) projections for 2 Circles Dataset with $d = 2$.

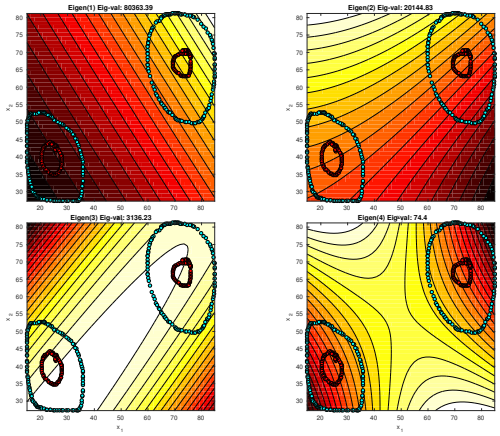

Figure 27: KPCA (Poly) Eigenvector Isolines for 2 Circles Dataset with $d = 2$

28 it's clear that we can achieve a subspace where the classes are well separated; e.g. with Eigenvector combination $\mathbf{v}^1 \times \mathbf{v}^3, \mathbf{v}^1 \times \mathbf{v}^3, \mathbf{v}^1 \times \mathbf{v}^8$. By taking a closer look into the isolines of those Eigenvectors we can see how this separation is achieved.

Figure 28: KPCA (RBF) projections for Loops Dataset with $\sigma = 20$.

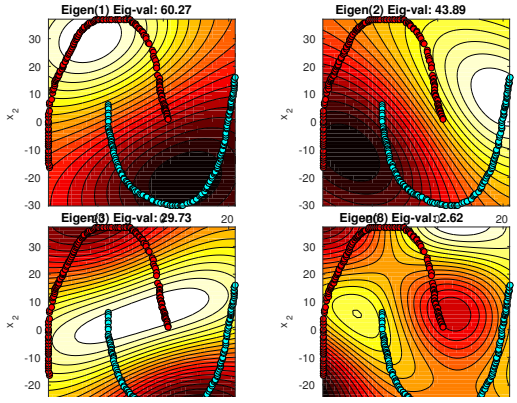

Figure 29: KPCA (RBF) Eigenvector Iso-lines for Loops Dataset with $\sigma = 20$.

4.3 Performance of PCA/KPCA on High-Dimensional Datasets

TASK 2: Compare PCA and Kernel PCA on High-Dimensional Data
You will now compare Kernel PCA and PCA on high-dimensional datasets. You will choose the hyper-parameters carefully and study whether you can achieve a good projection with each method. A good projection's definition then depends on the application:

In the case of kPCA as a pre-processing step for classification or clustering, a good projection is when the clustering/classification algorithms perform well. As we have not seen any classification or clustering method in class yet, the quality of the projection can be estimated visually with the visualizations tools provided by ML_toolbox, e.g. visualizing the projections on the Eigenvectors (PCA), the iso-lines of the Eigenvectors (kernel PCA) or for instance by estimating the separation between the labeled classes after projection.

1. **Breast Cancer Wisconsin:** This dataset is used to predict "benign" or "malignant" tumors. The dataset is composed of $M = 698$ datapoints of $N = 9$ dimensions, each corresponding to cell nucleus features in the range of $[1, 10]$. The datapoints belong to two classes $y \in \{\text{benign}, \text{malignant}\}$ (See Figure 30).

   HINT: Try to find a projection in which the data has few overlapping data-points from different classes.

   This dataset was retrieved from:
   `https://www.kaggle.com/uciml/breast-cancer-wisconsin-data`

2. **Ionosphere:** This radar data was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type

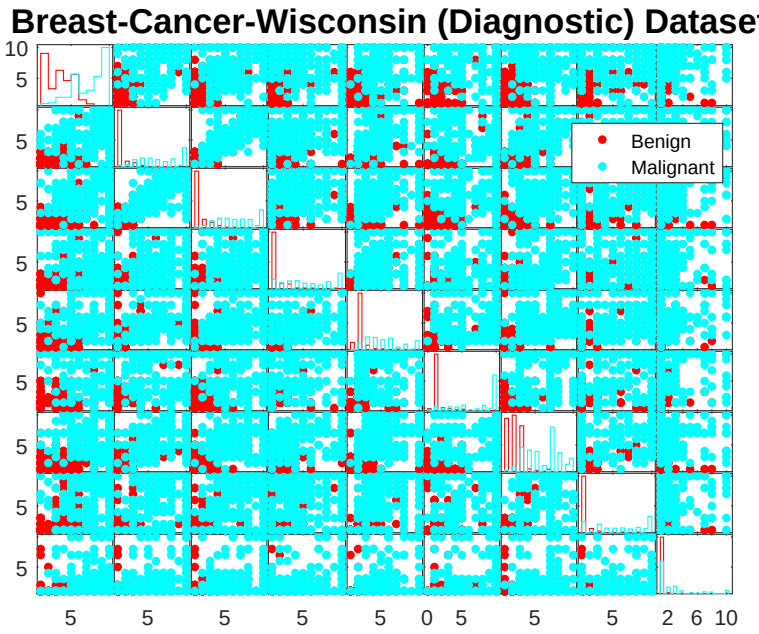

Figure 30: Scatter Matrix of Breast Cancer Wisconsin Dataset on Original Dimensions.

of structure in the ionosphere. "Bad" returns are those that do not; their signals pass through the ionosphere. The data is very noisy and seems to have a lot of overlapping values. The dataset is already normalized between -1 and 1.

> **HINT:** Try to find a projection in which the data has few overlapping data-points from different classes.

This dataset was retrieved from:

`https://archive.ics.uci.edu/ml/datasets/Ionosphere`

3. **Hayes-Roth:** This dataset is designed to test classification algorithms and has a highly non-linear class repartition. As seen in Figure 32, the classes of this dataset seem to be completely overlapping.

> **HINT:** Find good kernels and projections that will ease the task of separating all three classes.

This dataset was retrieved from:

`https://archive.ics.uci.edu/ml/datasets/Hayes-Roth`

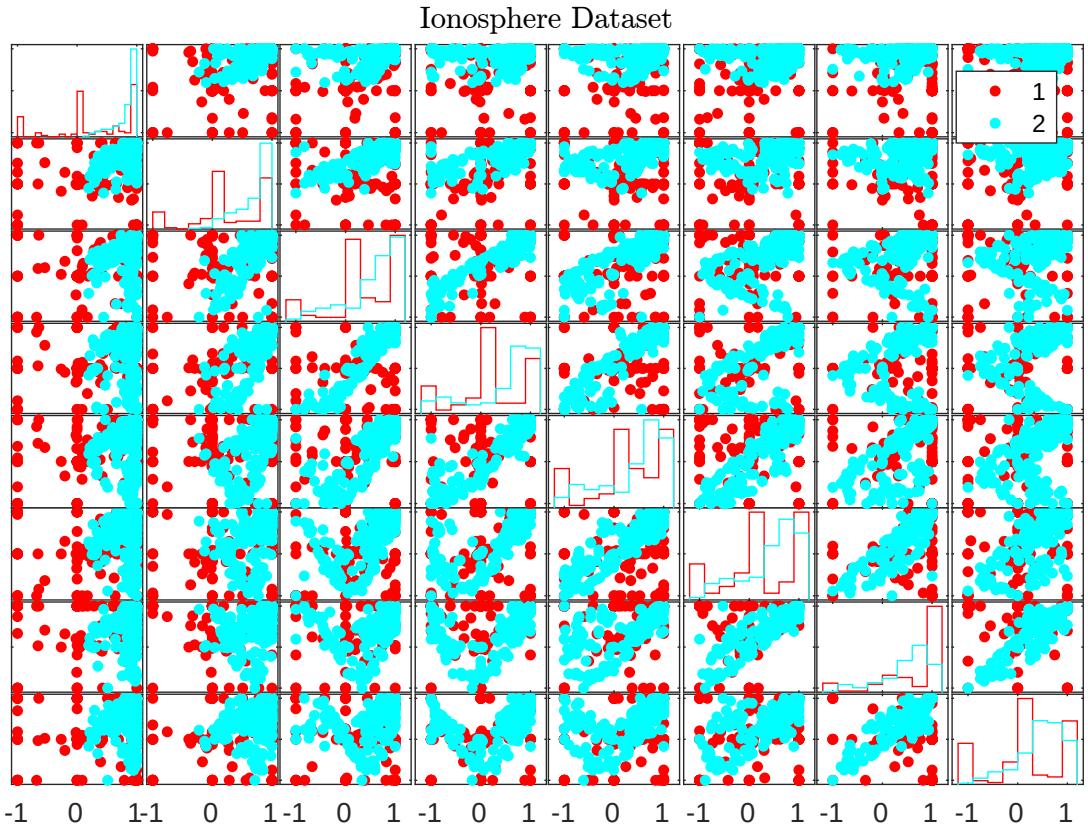

Figure 31: Scatter Matrix of Ionosphere Dataset on Original [1:4:32] Dimensions.

To load these datasets, we have provided example code in sub-block `1(a-c)` of the MATLAB script: `TP1_kPCA_HighD.m`. Visualizing the scatter matrix of a high-dimensional dataset can be computationally demanding, to this end, one can select which dimensions to visualize in the scatter matrix as follows:

```
% Plot original data
plot_options            = [];
plot_options.is_eig     = false;
plot_options.labels     = labels;
plot_options.title      = 'Ionosphere Dataset';

viz_dim = [1:4:32];

if exist('h1','var') && isvalid(h1), delete(h1);end
h1 = ml_plot_data(X(viz_dim,:)',plot_options);
```

The rest of the code blocks in the `TP1_kPCA_HighD.m` MATLAB script include the same functions used in `TP1_kPCA_2D.m` to compute the PCA and Kernel PCA projections.

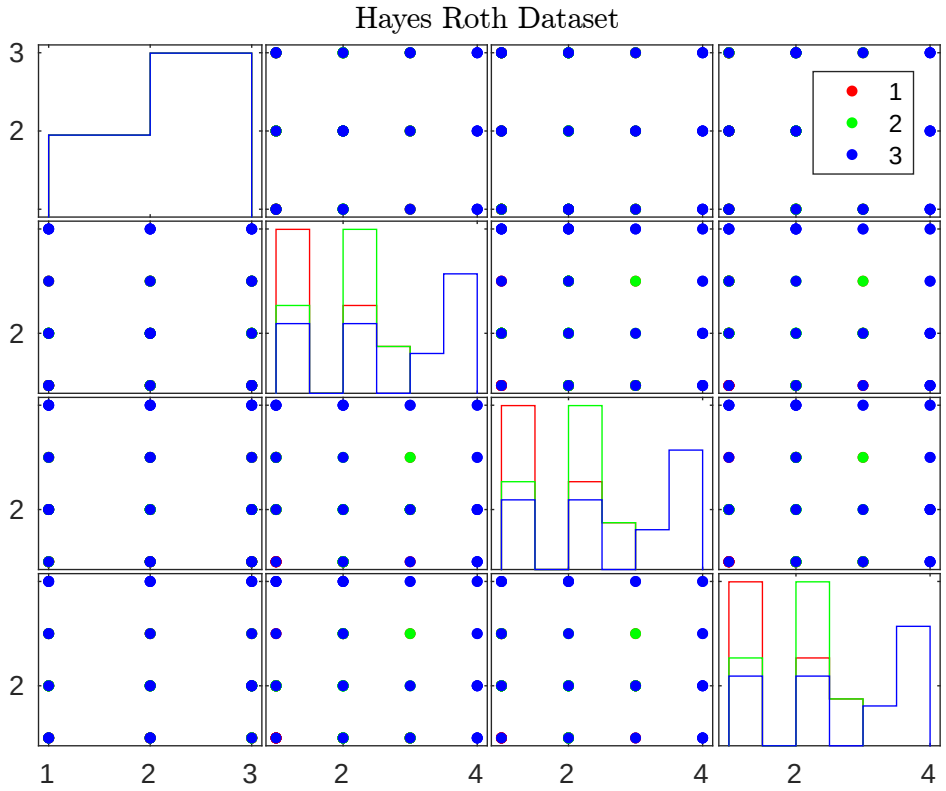

Figure 32: Scatter Matrix of Hayes Roth Dataset

HINT: To find a reasonable range for σ in the RBF Kernel one could analyze the Euclidean pair-wise distances with the following functions: `pdist.m` and `hist.m`.

Solution Task 2:

1. **Breast Cancer Wisconsin:** For the Breast Cancer Wisconsin Dataset applying PCA is sufficient to achieve a linearly seperable lower-dimensional embedding, as can be seen in 33, the class distributions seem to have been seperated by simple linear transformations.

   By analyzing the Eigenvalues in Figure 34, from component (eigenvector) number 2/3 and up, the eigenvalues are basically constant compared to the first ones, indicating that setting p to either 2 or 3 could be enough to represent the dataset correctly. From Figure 33 one can see that when $p = 2$ the datapoint form the two classes seem to be separated nicely, with some overlapping points. This can be better visualized in Figure 35 when $p = 3$. In fact, if we analyze the projection on the first eigenvector (in Figure 33) we can see this nice separation already. This can indicate that the first eigenvector explains much of the variance of the dataset and that it also encapsulates the correlations between most of the dimensions.

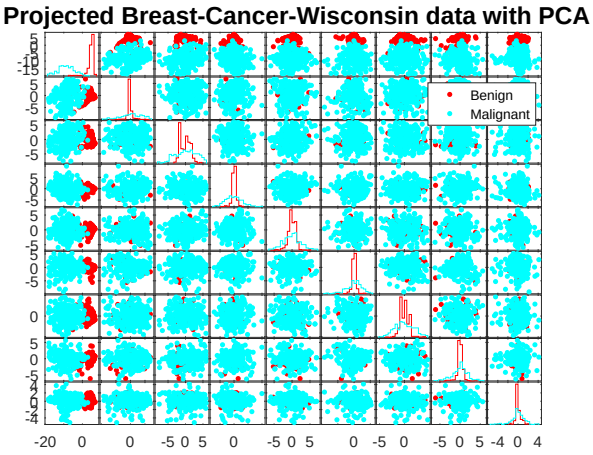

Figure 33: Scatter matrix of Breast Cancer dataset projected on each eigenvector..

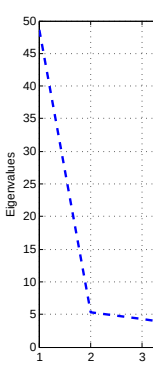

Figure 34: Plot of PCA Eigenvalues

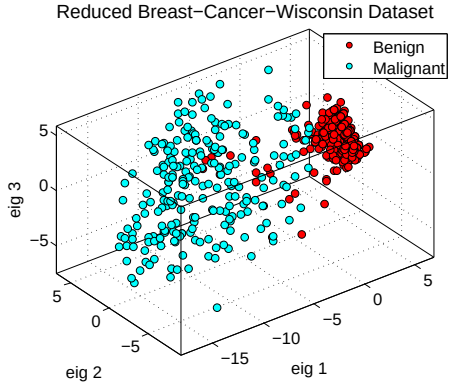

Figure 35: PCA Projected Dataset with p=3

For the sake of completeness, we try to find the optimal embedding with Kernel PCA as well. Let's assume the RBF is the best kernel for this dataset. Now we need to find the best kernel width σ, since this data is hard to visualize, one could compute a histogram of the pair-wise Euclidean distances $||\mathbf{x} - \mathbf{x}^i||$ between all data-points and find the minimum and maximum distances in the data. As see in Figure 36, the pair-wise Euclidean distances in the dataset seem to follow this distribution in the range of $0 - 25$. We thus, choose to do a grid search of $\sigma = [1, 50]$, which yields the Eigenvalue plots shown in Figure 37. Reasonable values are in the range of $5 < \sigma < 15$. Not surprisingly, by projecting the dataset with any of these values, say $\sigma = 8.8$ and $\sigma = 15$, we can almost achieve the same embedding as in PCA (see Figure 38, 39).

2. **Ionosphere:** This dataset seems to be non-linearly separable, hence, applying PCA will not achieve any good embedding, this can be seen in Figures 40 and 41. Even though this is a high-dimensional dataset, one can still visually extract characteristics of the class distibution by looking at the scattering matrix plots (Figure 31). For example, by looking at the first off-diagonal plot; i.e. x_1 vs x_2, we

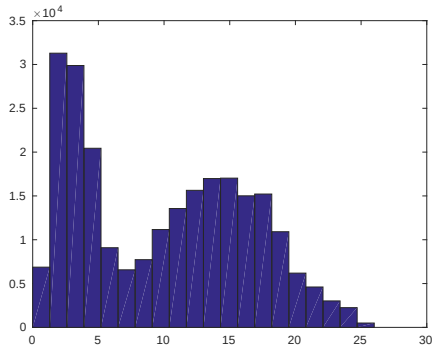

Figure 36: Pair-wise Euclidean distances of Breast Cancer Wisconsin Dataset

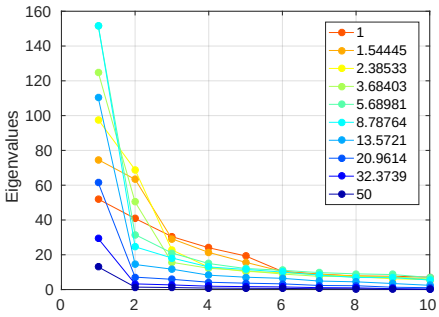

Figure 37: Grid Search for σ RBF Kernel PCA.

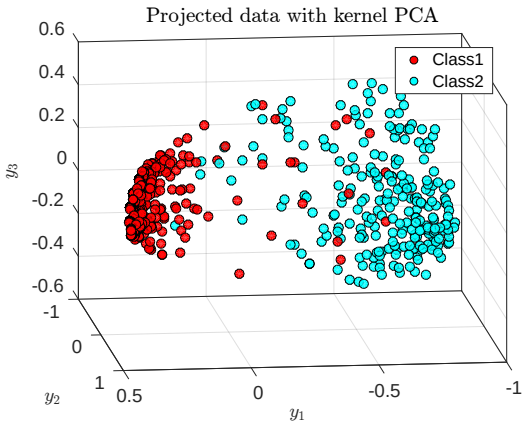

Figure 38: RBF Kernel PCA Projected Dataset with σ=8.8

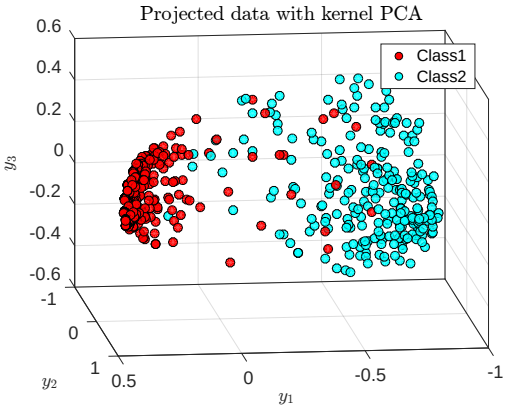

Figure 39: RBF Kernel PCA Projected Dataset with σ=15

can see that the blue class seems to always be clustered radially and surrounded by the red class. Therefore, an RBF kernel seems appropriate. The kernel width should then be chosen according to the scale of each cluster and the distance to the other cluster. The white cluster has a width of around 0.5, whereas the span of the red cluster is about 2. A kernel width of $\sigma = 1$ can thus be chosen as an initial guess. As can be seen in Figure 42, a good choice for p would be 3, the resulting projection can be seen in Figure 43.

Following the same procedure as in the Breast Cancer Dataset, we can compute the pairwise Euclidean distances. In this case, the histogram is quite revealing (see Figure 44), we can see two separate bimodal distributions and the range of Euclidean distances $[0, 8]$ is much higher than the scale of the data [-1, 1]. For this reason, we will do a grid search on the range of $\sigma = [0.5, 10]$. Figure 45 suggests that an optimal range of σ is between $[3, 10]$. By projecting the dataset with the $\sigma = 3$ can achieve a different embedding (see Figure 46, 47).

3. **Hayes-Roth:** This dataset is clearly non-linearly separable. Moreover, it is quite hard to visualize in its original feature space (Figure 32), since the points take

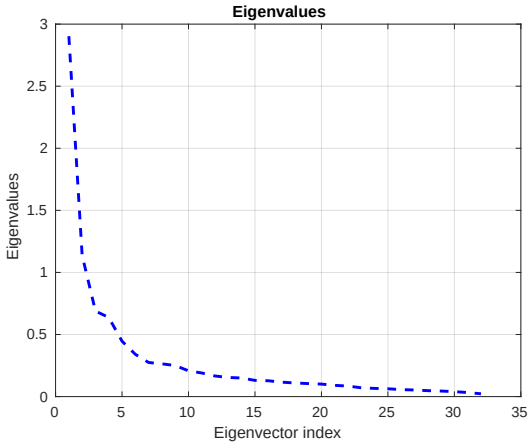

Figure 40: PCA Eigenvalues of Ionosphere Dataset

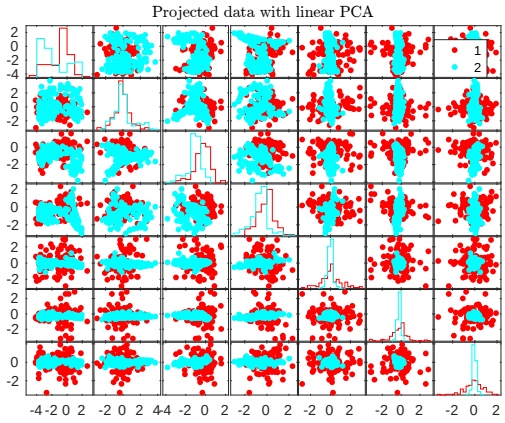

Figure 41: PCA projection of Ionosphere Dataset with $p = 7$.

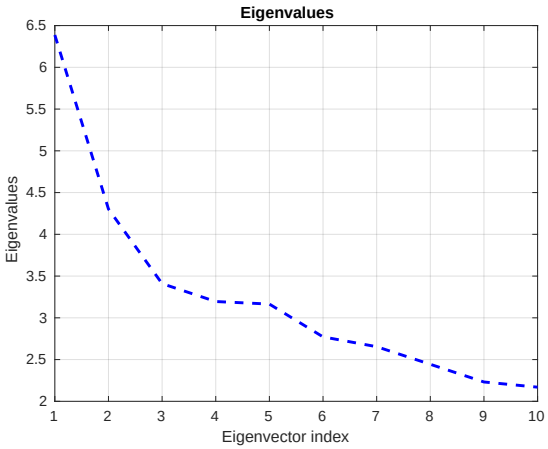

Figure 42: Kernel (RBF) PCA Eigenvalues of Ionosphere Dataset with $\sigma = 1$

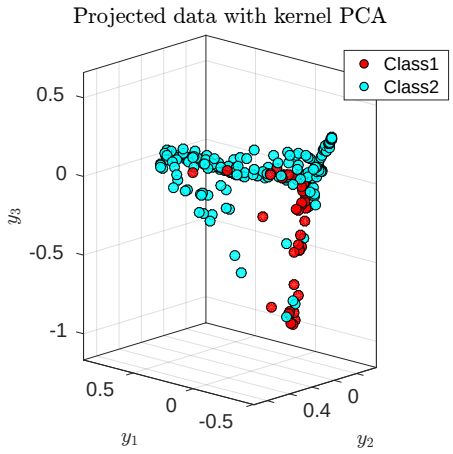

Figure 43: Kernel (RBF) PCA Projection of Ionosphere Dataset with $\sigma = 1$

discrete values and, thus, all points overlap on top of each other in the original dimensions. In such situations one could solve this problem in two ways:

Solution 1: Grid Search on the Hyper-parameters and find the optimal projection. To find a feasible range, we compute the pair-wise Euclidean distances and determine a "ball-park" estimate of the feasible range. Figure 48 shows the pair-wise distance, which fall in the range of $[0, 5]$, grid-search of the RBF Kernel was performed on the range $\sigma = [0.5, 10]$ (Figure 49). From these results we can see that some feasible σ values are in the range of $[1.3, 3.6]$. We choose $\sigma = 1.8$ and project the dataset on the first 5 principal components. As shown in 50, we can see that in some combination of Eigenvector Projections (e.g. $v_2 \times v_3$), the classes are somewhat visually separable.

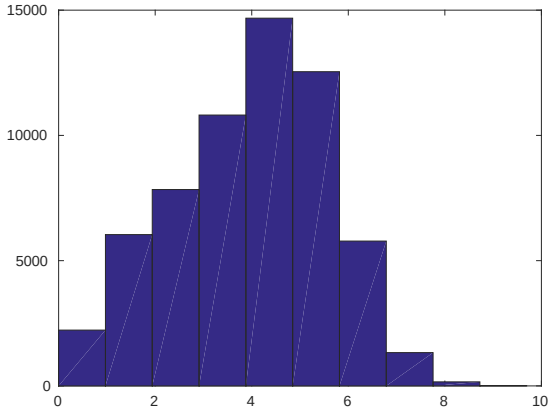

Figure 44: Pair-wise Euclidean Distances of Ionosphere Dataset.

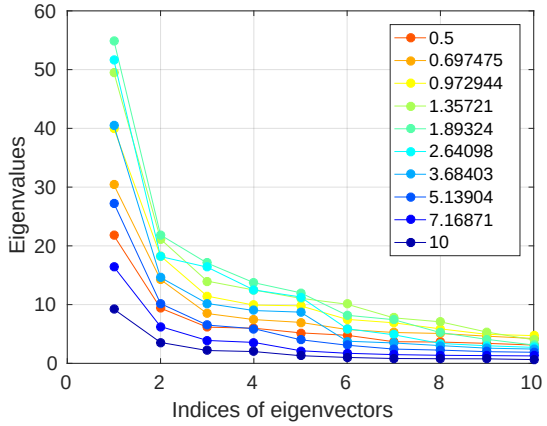

Figure 45: Grid Search for σ RBF Kernel PCA.

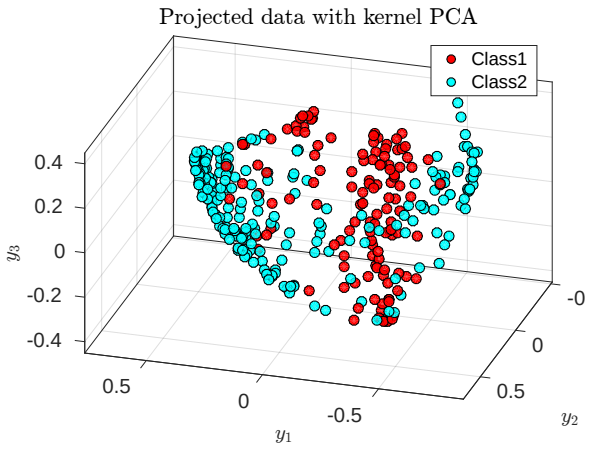

Figure 46: Kernel (RBF) PCA Eigenvalues of Ionosphere Dataset with $\sigma = 3$ and $p = 3$

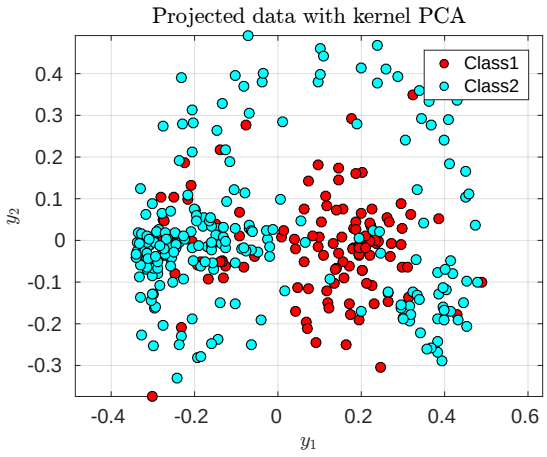

Figure 47: Kernel (RBF) PCA Eigenvalues of Ionosphere Dataset with $\sigma = 3$ and $p = 2$

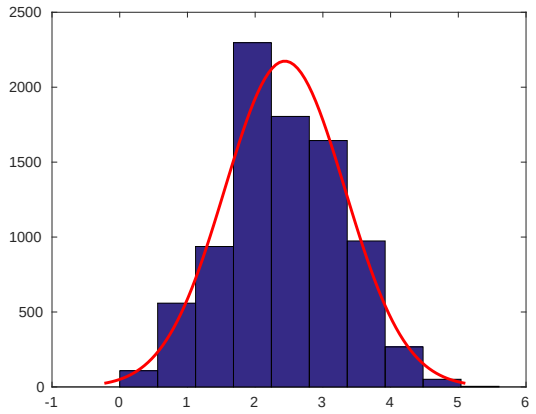

Figure 48: Pair-wise Euclidean Distances of Hayes-Roth Dataset.

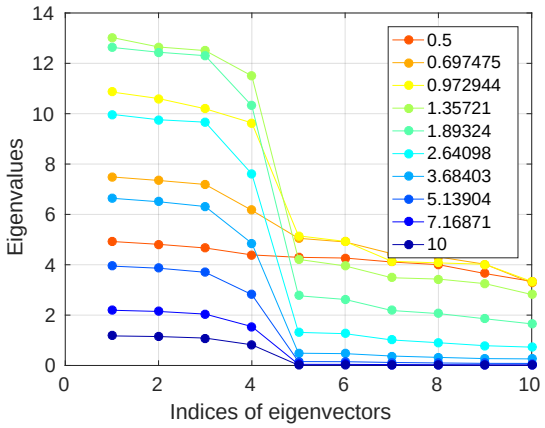

Figure 49: Grid Search for σ RBF Kernel PCA.

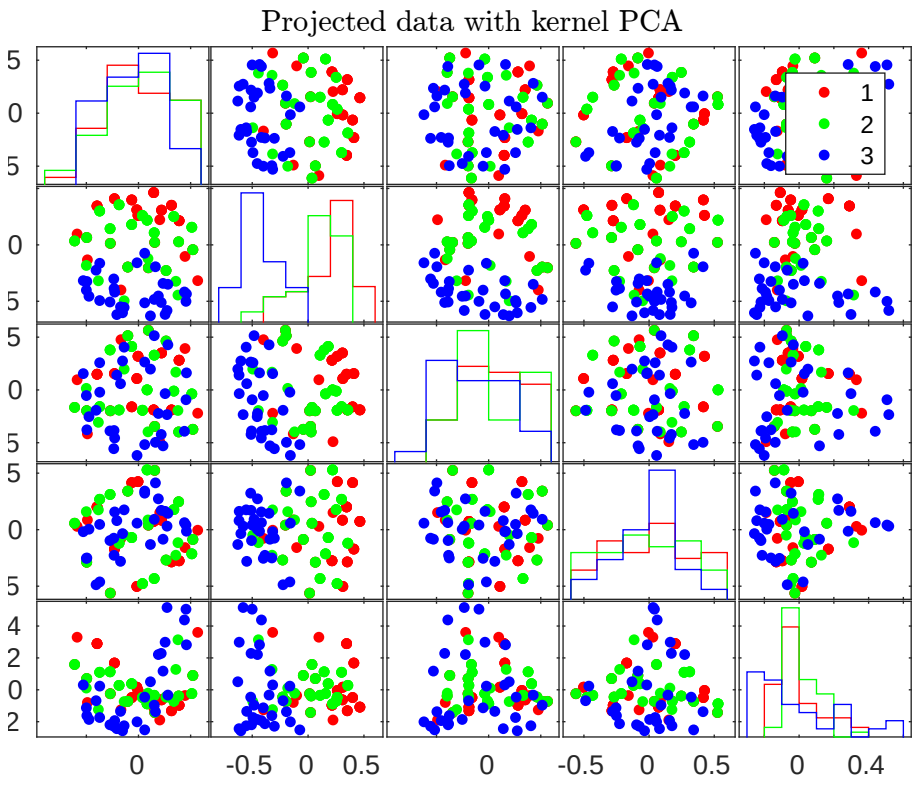

Figure 50: Scatter Matrix of PCA Projected Hayes Roth Dataset with RBF Kernel $\sigma = 1.8$

Solution 2: Another possibility is to use the linear PCA projections to see if some structure becomes apparent, as shown in Figure 51. Another advantage of PCA is that the data was initially on a 4*4 grid in the original dimensions and it becomes much more spread out and easier to read in the projection space. Hence, we can then apply Kernel PCA on the linearly projected space. By looking at the linear projection of the first two Eigenvectors, there seems to be a radial structure separating class 3 (blue) and classes 1-2 (red and green). We can quickly check that the blue class is easily separable from the other two classes with the simplest kernel: a degree $d = 2$ homogeneous kernel can separate the blue class from the other points. By applying such projection (Figure 52), we can see that indeed we can achieve a less discretized and more separable dataset.

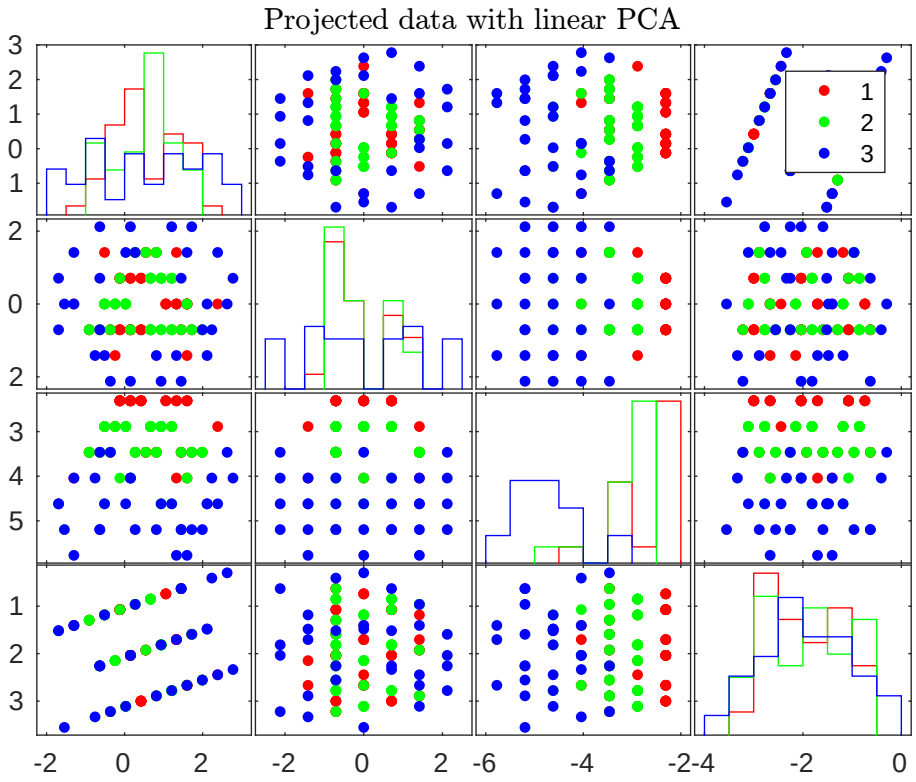

Figure 51: Scatter Matrix of PCA Projected Hayes Roth Dataset

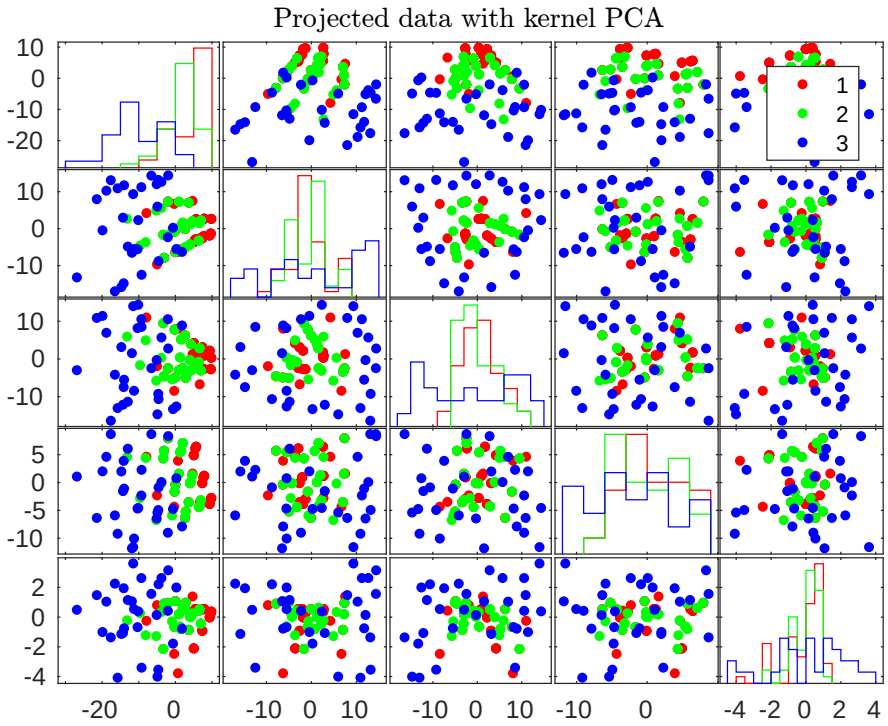

Figure 52: Scatter Matrix of Kernel PCA Projected Hayes Roth Dataset with Polynomial Kernel $d = 2$

References

[1] Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. Advances in kernel methods. chapter Kernel Principal Component Analysis, pages 327–352. MIT Press, Cambridge, MA, USA, 1999.

[2] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.