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

Professor: Aude Billard
Assistants: Bernardo Fichera, Mikhail Koptev, Arnaud Guibbert
E-mails: aude.billard@epfl.ch,
bernardo.fichera@epfl.ch, mikhail.koptev@epfl.ch, arnaud.guibbert@epfl.ch

Spring Semester 2022

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-3D.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 $f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^N \to \mathbf{y} \in \mathbb{R}^p$ 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 $f(\mathbf{x}) : \mathbb{R}^N \to \mathbb{R}^{p\leq N}$ 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 $\mathbf{V} = [\mathbf{v}^1, \dots, \mathbf{v}^N]$. 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 equation (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:** $k(\mathbf{x}, \mathbf{x}^i) = (\langle \mathbf{x}, \mathbf{x} \rangle + c)^d$,
  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-3D.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.

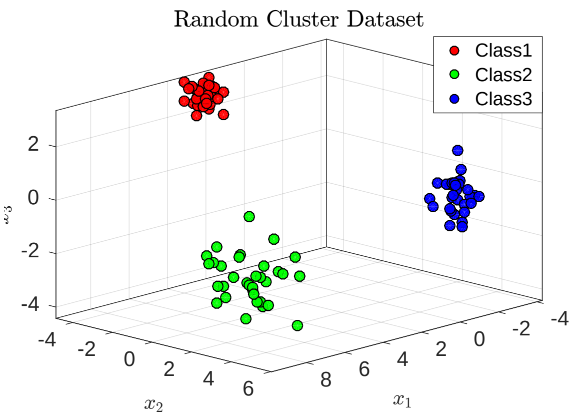

Figure 1: 3D Random Clusters Dataset.

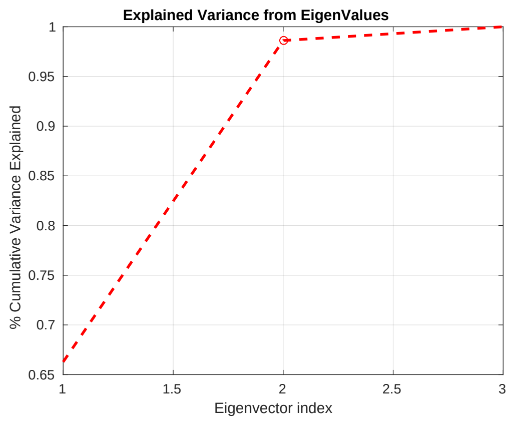

Figure 2: Explained Variance plot of 3D Random Clusters Dataset; $p = 2$ for $\sigma > 90\%$.

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. If 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.

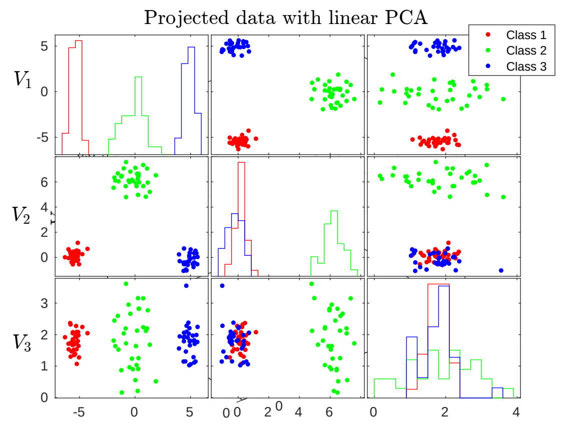

Figure 3: Principal Component Analysis (PCA) of 3D Random Clusters Dataset with $p = 3$.

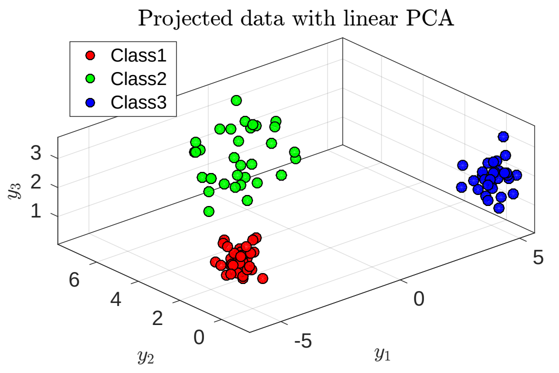

Figure 4: Projected Data of 3D Random Clusters Dataset PCA with $p = 3$.

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 102, as:

```
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.

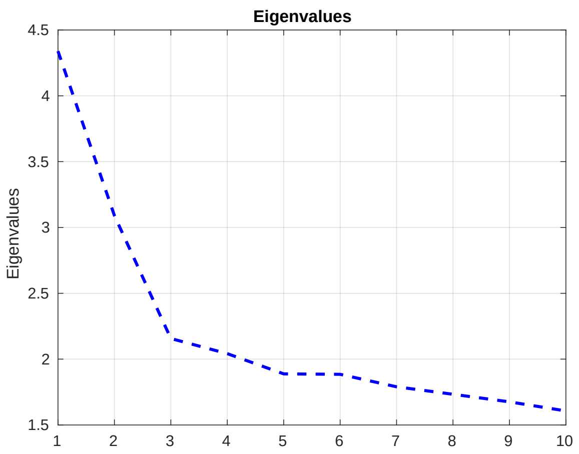

Figure 5: Eigenvalues of kPCA from 3D Random Clusters Dataset.

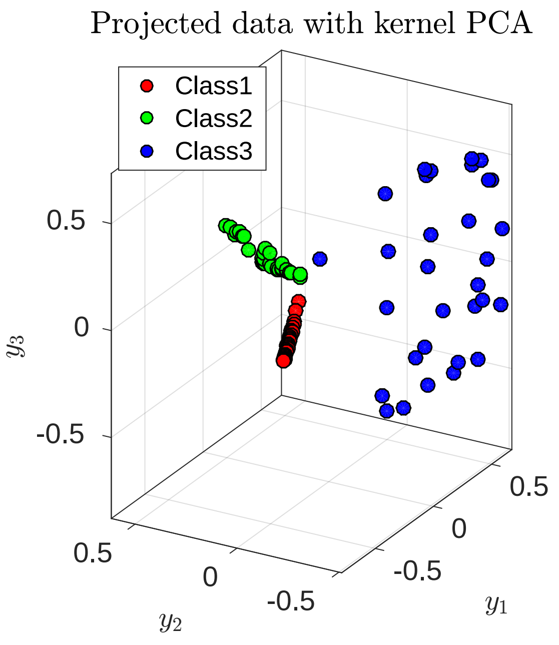

Figure 6: Projected Data of 3D Random Clusters Dataset kPCA with $p = 3$.

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:

```
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:

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

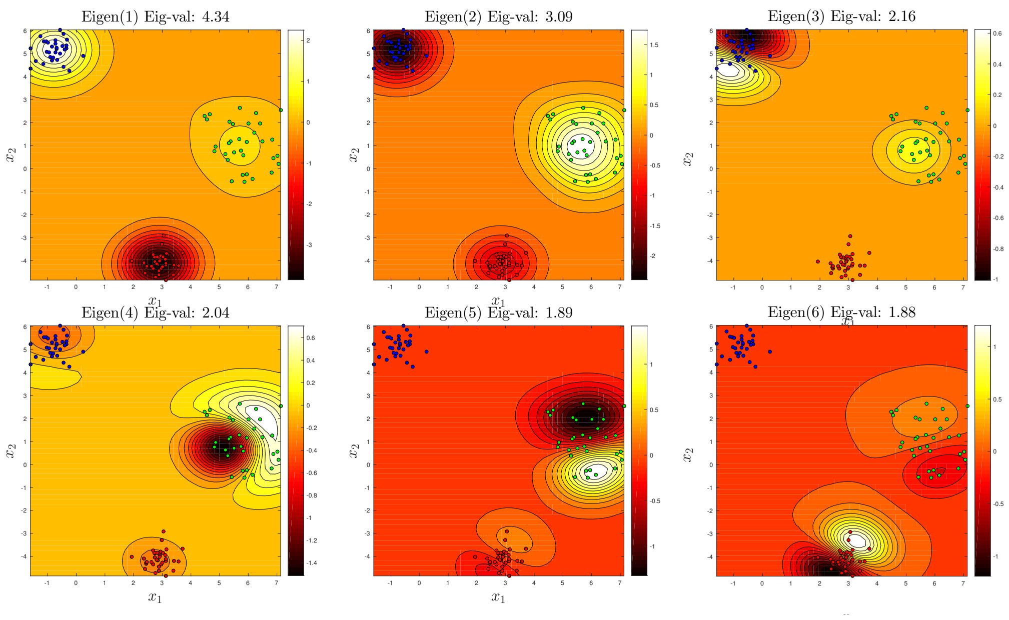

Figure 7: Isolines of the first $p = 6$ Eigenvectors of kPCA from the 3D Random Clusters Dataset (showing only the first 2 dimensions of the original data).

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-3D.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';

% Set Range of Hyper-Parameters
kpars = [0.001,0.01, 0.05,0.1,0.2, 0.5, 1, 2];
% kpars = [0,0,0,0; 1,2,3,4]; % for 'poly': [offset; order]
[ eigenvalues ] = ml_kernel_grid_search(X',grid_options,kpars);
```

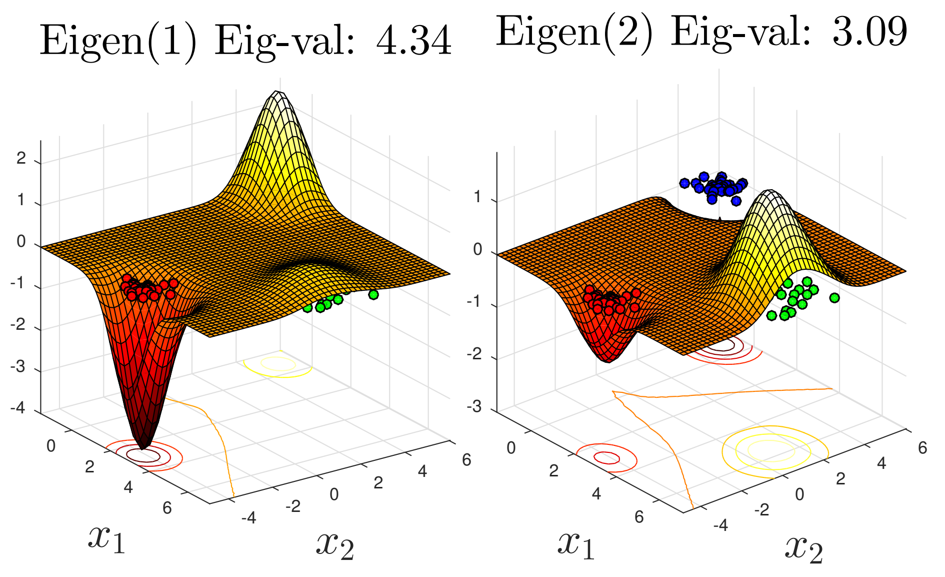

Figure 8: Isolines (Surface plot) of the first $p = 2$ Eigenvectors of kPCA from the 3D Random Clusters Dataset.

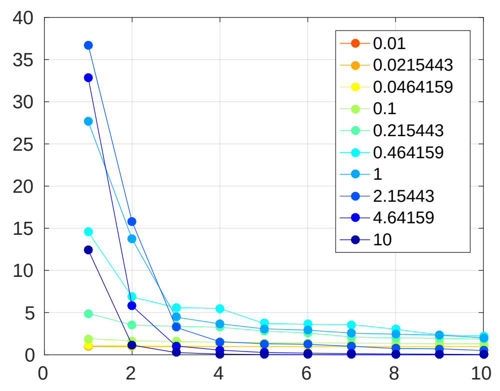

Figure 9: Grid Search of σ for the RBF Kernel.

4.2 Analysis of Kernel Choice and Hyper-parameters

TASK 1: Try Kernel PCA on Non-Linearly Separable 2D/3D 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.

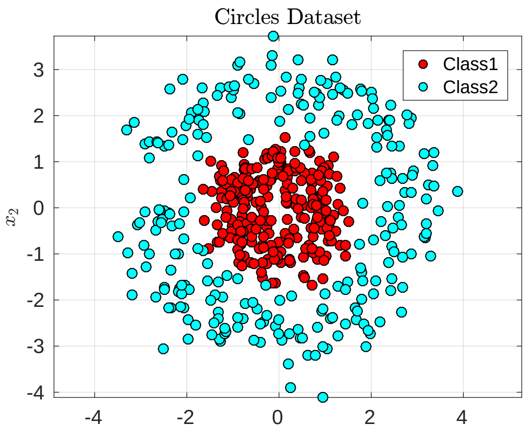

Figure 10: 2D Concentric Circles Dataset.

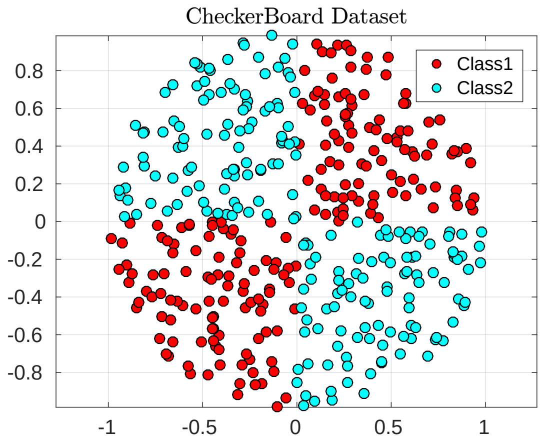

Figure 11: 2D CheckerBoard Dataset.

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).

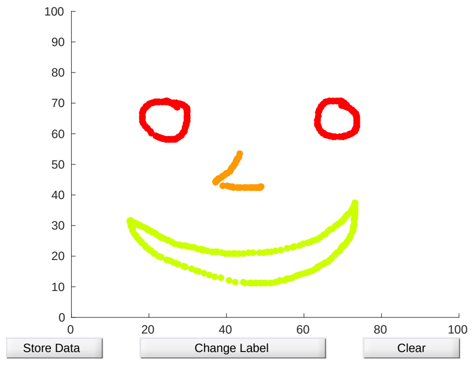

Figure 12: ML_toolbox 2D Data Drawing GUI.

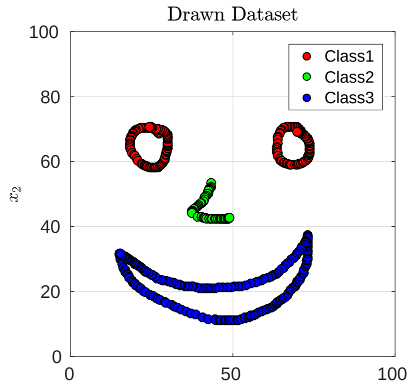

Figure 13: Data After Storing it in MATLAB workspace.

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

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 14).

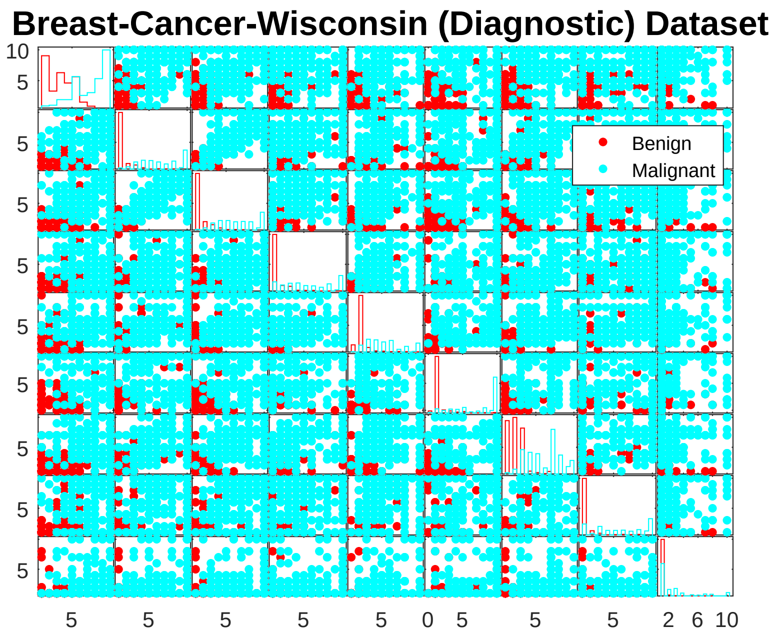

Figure 14: Scatter Matrix of Breast Cancer Wisconsin Dataset on Original Dimensions.

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 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.

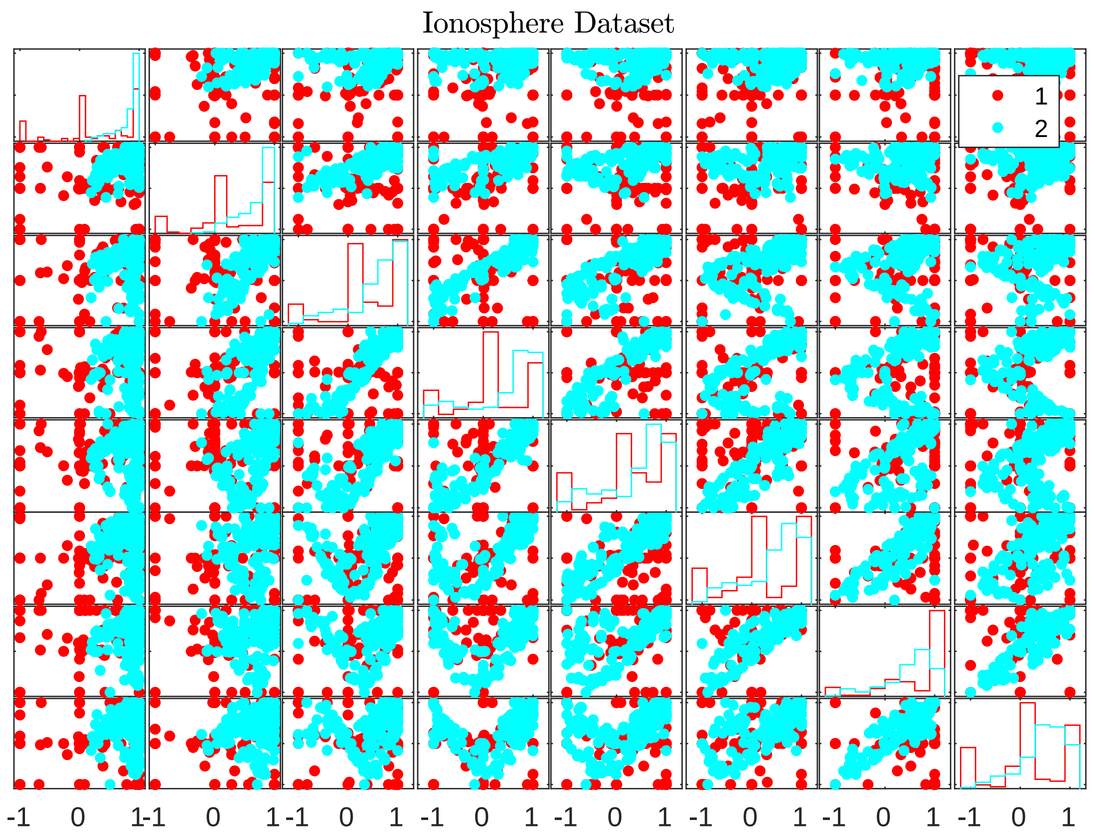

Figure 15: Scatter Matrix of Ionosphere Dataset on Original [1:4:32] Dimensions.

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 16, the classes of this dataset seem to be completely overlapping.

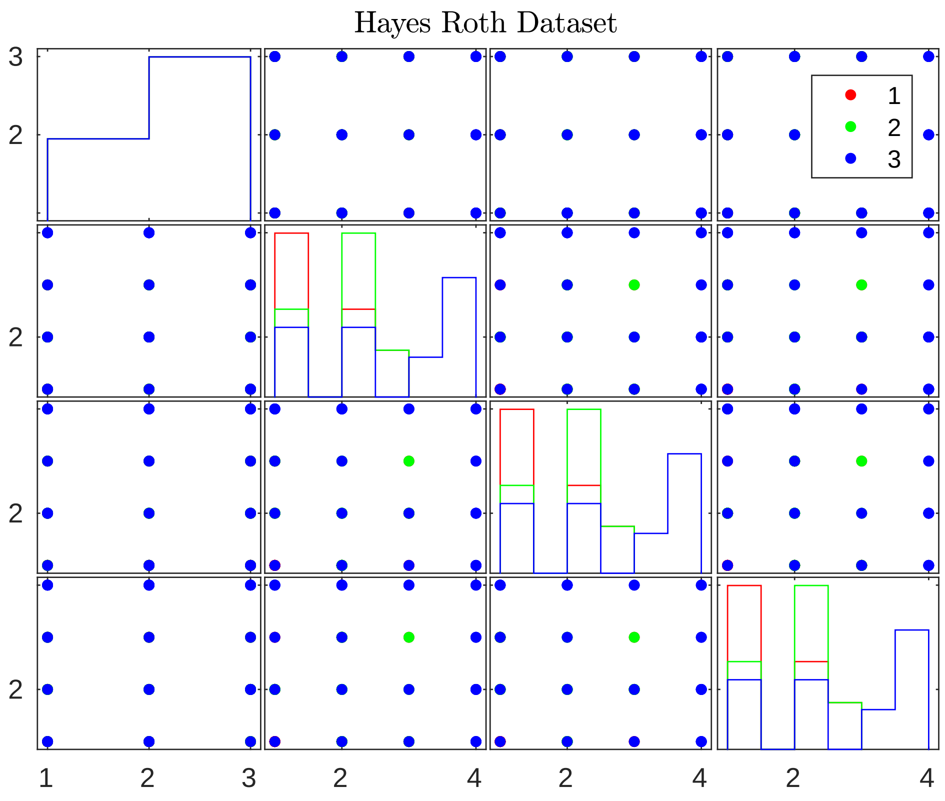

Figure 16: Scatter Matrix of Hayes Roth Dataset

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`

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-3D.m` to compute the PCA and Kernel PCA projections.

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

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.