University of Khartoum

Faculty of Engineering

Department of Electrical and Electronics Engineering

DSP Mini Project Report

Feature Extraction Using Kernel-PCA

Prepared By:

Hussam Eldeen Mohammed Abdulajabar - 144024
Mohammed Kamal Ahmed Gumma - 144070
Mohammed Ali Abdallah Ahmed - 144080
Salah Mahadi Abdel-fatah Abdel-salam - 144035
Mohammed Salah Ibrahim Abdallah - 144074

April 4th 2018

1. Introduction

Data mining is concerned with finding meaningful patterns in large sets of data. The swift development in computation power that has taken place in recent years has increased the amount of available data in an avalanche-like fashion (consider for example the Human Genome Project, in which the whole human DNA is stored in a database). The task of analyzing these gigantic amounts of data can appear overwhelming. Naturally, there is a need for tools that can aid the analyst in identifying the interesting features of the data. In the past decades, a group of methods known as dimension reduction methods have gained wide-spread attention. As the name implies, dimension reduction methods seek to solve this problem by reducing the number of dimensions of the data, while retaining most of the information in the data set. By removing redundant dimensions, it becomes easier to recognize patterns or tendencies in the remaining data.
In this report, one particular dimension reduction method is considered, namely Kernel Principal Component Analysis (KPCA). The method is implemented in MATLAB and applied to data sets. In addition, different ways of modifying kernel PCA in order to improve the results for these particular datasets will be considered.

2. State of the Art

The outline of the report provides the description of kernel PCA and considers other techniques that are used in the application of the method namely the centering in feature space. And the numerical experiments that were performed on an iris dataset. The results are commented along with the discussion.
Kernel PCA uses the same basic idea as the statistical method Principal Component Analysis (PCA), namely that it seeks to project the set of data onto a low-dimensional subspace that captures the highest possible amount of variance in the data. However, while PCA performs a linear separation of the data in the original space (referred to as Rd), kernel PCA embeds the data into a high dimensional space, called the feature space, by a mapping function Φ and performs a linear separation in that space instead. In this way also nonlinear separations of data are made possible. Thus, while kernel PCA looks for linear features in feature space, these features correspond to nonlinear features in the original lower dimensional space Rd. The data in the feature space is projected onto a low-dimensional subspace, spanned by the eigenvectors that capture most of the variance. One important fact is that it is not necessary to know the mapping Φ nor the feature space explicitly in order to perform kernel PCA. Instead computations are performed on the inner product of pairs of points which are stored in a kernel matrix. The procedure of working with the data in feature space without knowing the mapping Φ nor F is known as ”the kernel trick” and is a central part of the kernel PCA method. In the original space, the two types of data points cannot be linearly separated. However, by extracting nonlinear features, the points can be sorted according to their class.

3. Problem Statement

In terms of performance, having data of high dimensionality is problematic because it’ll result a high computational cost to perform machine learning & inference and it often leads to over-fitting that’s why dimensionality reduction is important.

4. Objectives

1. Visualization: projection of high-dimensional data onto 2D or 3D.
2. Data compression: efficient storage and retrieval.
3. Machine Learning Performance Enhancement.

5. Significance of the project

Feature extraction is a general term for methods of constructing combinations of the variables to get around the large memory and computation power problems in case of complex datasets while still describing the data with sufficient accuracy and effective model construction.

Advantages:

1. It reduces the time and storage space required.
2. Improves the performance of the machine learning model and reduces the computational power.
3. It becomes easier to visualize the data when reduced to very low dimensions such as 2D or 3D.

6. Project Approach

Mathematical Formulation

Let the set of n data points be represented by

$$X = \{x_1, \ldots, x_n\}, \tag{1}$$

where $x_i \in R^d$, $i = 1, \ldots, n$. It is assumed that the data in X is centered in the original space R^d, so that

$$\sum_{i=1}^{n} x_i = 0. \tag{2}$$

This condition can be met by using the formula

$$\tilde{x}_k = x_k - \frac{1}{n} \sum_{i=1}^{n} x_i, \tag{3}$$

where $\tilde{x}_k$ is the k:th data point in the new, centered set.

The following is a description of how kernel PCA is performed on a data set X, consisting of n data points.

Let Φ be a mapping from the original space R^d of the data vectors, into a inner product space F, so that

$$\Phi : R^d \to F, \tag{4}$$

where the space F is referred to as the feature space. Thus, $\Phi(x_i)$ represents the image of the data vector $x_i \in R^d$ in feature space.

Let the kernel matrix K be a symmetric and positive semidefinite $n \times n$-matrix, with its elements defined by the inner product of all pairs of points $\Phi(x_i)$ and $\Phi(x_j)$ in feature space so that

$$K_{ij} \triangleq (\Phi(x_i) \cdot \Phi(x_j)), \quad i, j = 1, \ldots, n. \tag{5}$$

Interestingly, it is not necessary to known the values of $\Phi(x_i)$ and $\Phi(x_j)$ explicitly in order to compute the kernel matrix K. In other words, in order to map the data points into F, the mapping Φ is not required to be known. Instead, the elements of K can be calculated by computing the pairwise relation of all points $x_i, x_j \in X$ in the original space R^d.

This is motivated by Mercer's Theorem (see below) and done through a kernel function $k(x_i, x_j)$, so that

$$K_{ij} = k(x_i, x_j), \quad i, j = 1, \ldots, n. \tag{6}$$

This way the features space F and the mapping Φ are implicitly defined by the kernel function that is used. This is known as "the kernel trick" and is the perhaps most central part of the kernel PCA method.

There are a variety of kernel functions that can be used for kernel PCA. In this report the Gaussian radial basis function (RBF) has been used (equation (12)).

The data points represented in the kernel matrix K are assumed to be centered in feature space, in the sense that

$$\sum_{i=1}^{n} \Phi(x_i) = 0. \tag{7}$$

For a noncentered set, this centering can be achieved by the inner product computation

$$(\tilde{K})_{ij} = ((\Phi(x_i) - \overline{\Phi}) \cdot (\Phi(x_j) - \overline{\Phi})) \tag{8}$$

where $\tilde{K}$ is the centered kernel matrix and $\overline{\Phi}$ is the center point in the feature space, given by

$$\overline{\Phi} = \frac{1}{n} \sum_{i=1}^{n} \Phi(x_i). \tag{9}$$

A derivation of the centering in feature space is found in the appendix.

The eigenvectors of the centered kernel matrix $\tilde{K}$ can be obtained by solving the equation

$$\tilde{K}V = \Lambda V, \tag{10}$$

where the columns of V represent the normalized eigenvectors and Λ is a matrix in which the diagonal consist of the corresponding eigenvalues and all other elements in Λ are zero. The eigenvectors in V are assumed to be ordered according to the size of their corresponding eigenvalues, in descending order. Also the eigenvalues in Λ are sorted in descending order.

The representation of the low-dimensional points $\{\chi_1, \ldots, \chi_n\}$, with $\chi_i \in R^l$, $i = 1, \ldots, n$, can be obtained by projecting the kernel matrix $\tilde{K}$ onto the l eigenvectors that have the largest corresponding eigenvalues, by using the formula

$$\chi_i = \tilde{K}_i V^l, \tag{11}$$

where $\tilde{K}_i$ is the i:th row of K, and V^l is the matrix consisting of the l first eigenvector columns in V.

The kernel matrix is defined as (5) and thus contains the inner product of each pair of points x_i, x_j in feature space F. However, the value of K_{ij} is determined by some kernel function $k(x_i, x_j)$. This is a consequence of Mercer's Theorem, which states that any symmetric, positive semidefinite matrix can be regarded as a kernel matrix, that is as an inner product matrix in some space, which in the case of kernel PCA is the feature space.

If k is a continuous kernel of a positive integral operator, then it is possible to construct a mapping into a space where k acts as a dot product.

The implication of Mercer's theorem for kernel PCA is that the inner product $(\Phi(x_i) \cdot \Phi(x_j))$ can be substituted by the kernel function $k(x_i, x_j)$ that we choose. There are many different kernel functions available, for example the Gaussian radial basis function [1]

$$K_{ij} = k(x_i, x_j) = \exp(-\sigma \|x_i - x_j\|^2), \tag{12}$$

where σ is a parameter. As can be seen from equation (12), K_{ij} tends to go toward zero if the points x_i and x_j are far away from each other in R^d, while K_{ij} is close to one as the two points are near each other in R^d.

As seen from the definition in equation (5), K_{ij} is the inner product of $\Phi(x_i)$ and $\Phi(x_j)$. Thus, the following relationship between the points x_i, x_j in the original space R^d and their images $\Phi(x_i)$, $\Phi(x_j)$ in the feature space is valid:

If $K_{ij} \approx 0$, i.e. x_i and x_j are distant from each other in R^d, then $\Phi(x_i)$ and $\Phi(x_j)$ are orthogonal in feature space. Likewise, if $K_{ij} \approx 1$, i.e. x_i and x_j are near each other in R^d, then $\Phi(x_i)$ and $\Phi(x_j)$ are parallel (and if $\Phi(x_i)$ and $\Phi(x_j)$ are assumed to be normal vectors in F, the two points are equal to each other).

The consequence of this is that data points from a high-dimensional data set X that are geometrically close to each other in the space R^d, will be "almost" parallel in feature space. Furthermore, if the points in feature space are normal vectors in the sense that

$$\|\Phi(x_i)\| = 1, \quad i = 1, \ldots, n, \tag{13}$$

then points that are almost parallel will be be close to each other, as they are located on the hypersphere of radius one in the feature space.

Centering in feature space

In this section the formula for centering the kernel matrix will be derived. The derivation in this section is based on [13]. The centered kernel matrix $\tilde{K}$ is to be expressed in terms of the noncentered kernel matrix K.

A set of n centered data points in feature space can be written as

$$\tilde{\Phi}(x_i) \triangleq \Phi(x_i) - \frac{1}{n}\sum_{k=1}^{n}\Phi(x_k), \tag{34}$$

for $i = 1, \ldots, n$.

The centered kernel matrix is then defined as

$$\tilde{K}_{ij} = (\tilde{\Phi}(x_i) \cdot \tilde{\Phi}(x_j)). \tag{35}$$

By inserting (34) into (35), the centered kernel matrix can be expressed in terms of $\Phi(x_i)$:

$$\begin{aligned}\tilde{K}_{ij} &= (\Phi(x_i) - \frac{1}{n}\sum_{k=1}^{n}\Phi(x_k)) \cdot (\Phi(x_j) - \frac{1}{n}\sum_{l=1}^{n}\Phi(x_l)) \\ &= \Phi(x_i)\cdot\Phi(x_j) - \Phi(x_i)\cdot\frac{1}{n}\sum_{l=1}^{n}\Phi(x_l) - \frac{1}{n}(\sum_{k=1}^{n}\Phi(x_k))\cdot\Phi(x_j) \\ &+ \frac{1}{n^2}\sum_{k=1}^{n}\Phi(x_k)\sum_{l=1}^{n}\Phi(x_l).\end{aligned}$$

Rewriting the equation above in matrix form gives

$$\begin{aligned}\tilde{K}_{ij} &= K_{ij} - \frac{1}{n}\sum_{k=1}^{n}1_{ik}K_{kj} - \frac{1}{n}\sum_{l=1}^{n}K_{il}1_{lj} + \frac{1}{n^2}\sum_{k,l=1}^{n}1_{ik}K_{kl}1_{lj} \\ &= (K - 1_nK - K1_n + 1_nK1_n)_{ij},\end{aligned}$$

where 1_{ij} represents the element of an $n \times n$ matrix in which each element equals one, and 1_n is an $n \times n$ matrix in which each element equals $1/n$. Thus, the expression for centering the noncentered kernel matrix is

$$\tilde{K} = K - 1_nK - K1_n + 1_nK1_n. \tag{36}$$

For the test kernel matrix the derivation of the formula used for centering is performed in a similar way, namely by starting out from the definition of the noncentered test kernel matrix

$$K^{tt}_{ij} = (\Phi(x^{tt}_i) \cdot \Phi(x^{tr}_j)), \tag{37}$$

and the centered test kernel matrix

$$\tilde{K}^{tt}_{ij} = (\tilde{\Phi}(x^{tt}_i)) \cdot (\tilde{\Phi}(x^{tr}_j)), \tag{38}$$

where K^{tt} and $\tilde{K}^{tt}$ are $n_{tt} \times n_{tr}$ matrices and $i = 1, \ldots, n_{tt}$ and $j = 1, \ldots, n_{tr}$. Then,

$$\begin{aligned}\tilde{K}_{ij}^{tt} &= (\Phi(x_i^{tt}) - \frac{1}{n_{tr}} \sum_{k=1}^{n_{tr}} \Phi(x_k^{tr})) \cdot (\Phi(x_j^{tr}) - \frac{1}{n_{tr}} \sum_{l=1}^{n_{tr}} \Phi(x_l^{tr})) \\ &= \Phi(x_i^{tt}) \cdot \Phi(x_j^{tr}) - \frac{1}{n_{tr}} \Phi(x_i^{tt}) \cdot (\sum_{l=1}^{n_{tr}} \Phi(x_l^{tr})) - \frac{1}{n_{tr}} (\sum_{k=1}^{n_{tr}} \Phi(x_k^{tr})) \cdot \Phi(x_j^{tr}) \\ &+ \frac{1}{n_{tr}^2} (\sum_{k=1}^{n_{tr}} \Phi(x_k^{tr})) \cdot (\sum_{l=1}^{n_{tr}} \Phi(x_l^{tr})).\end{aligned}$$

Here, the term

$$\begin{aligned}\Phi(x_i^{tt}) \cdot (\sum_{l=1}^{n_{tr}} \Phi(x_l^{tr})) &= \Phi(x_i^{tt}) \cdot (\Phi(x_1^{tr}) + \cdots + \Phi(x_{n_{tr}}^{tr})) \\ &= \sum_{l=1}^{n_{tr}} \Phi(x_i^{tt}) \cdot \Phi(x_l^{tr}) \\ &= \sum_{l=1}^{n_{tr}} K_{il}^{tt} = \sum_{l=1}^{n_{tr}} K_{il}^{tt} 1_{lj}.\end{aligned}$$

Likewise, the other terms in the equation can be rewritten in a similar way, leading to

$$\begin{aligned}\tilde{K}_{ij}^{tt} &= K_{ij}^{tt} - \frac{1}{n_{tr}} \sum_{l=1}^{n_{tr}} K_{il}^{tt} 1_{lj} - \frac{1}{n_{tr}} \sum_{k=1}^{n_{tr}} 1_{ik} K_{kj}^{tr} + \frac{1}{{n_{tr}}^2} \sum_{k,l=1}^{n_{tr}} 1_{ik} K_{kl}^{tr} 1_{lj} \\ &= (K^{tt} - K^{tt} 1_{n_{tr}} - 1_{n_{tt}} K^{tr} + 1_{n_{tt}} K^{tr} 1_{n_{tr}})_{ij},\end{aligned}$$

where $1_{n_{tt}}$ is an $n_{tt} \times n_{tr}$ matrix in which each element equals $1/n_{tr}$ and $1_{n_{tr}}$ is an $n_{tr} \times n_{tr}$ in which each element equals $1/n_{tr}$. The formula for centering the test kernel matrix is then

$$\tilde{K}^{tt} = K^{tt} - 1_{n_{tt}} K^{tr} - K^{tt} 1_{n_{tr}} + 1_{n_{tt}} K^{tr} 1_{n_{tr}}. \tag{39}$$

Source Code

In this section, parts of the MATLAB-source code that was used in the experiments are included.

The code presented in this section is the main program, used for running the software. This function depends on the other functions either directly or indirectly.

In this section we project the original data in the original space.

```
addpath('../code');
load iris.data;
d=2;


%% original data
figure;hold on;
plot3(data(1:2:end,1),data(1:2:end,2),data(1:2:end,3),'b*');
plot3(data(2:2:end,1),data(2:2:end,2),data(2:2:end,3),'ro');
legend('class 1','class 2');
axis equal;
xlabel('x');
ylabel('y');
zlabel('z');
axis([-110 110 -110 110 -110 110]);
title('original data');
drawnow;
```

Gaussian Kernel Method & dimensionally reduced dataset projection

```
sigma = 0.02;
disp('Performing Gaussian kernel PCA...');
[Y1, eigVector]=kPCA(data,d,sigma); %Y1 is the dimentionality
reduced dataset using gaussian KPCA
figure;hold on;
plot(Y1(1:2:end,1),Y1(1:2:end,2),'b*');
plot(Y1(2:2:end,1),Y1(2:2:end,2),'ro');
legend('class 1','class 2');
title('Gaussian kernel PCA');
drawnow;
```

KPCA Function

```
function [Y, eigVector, eigValue]=kPCA(X,d,sigma)
%% kernel PCA
K0=kernel(X,sigma);
oneN=ones(N,N)/N;
K=K0-oneN*K0-K0*oneN+oneN*K0*oneN;
%% eigenvalue/eigen vector formulation
[V,D]=eig(K); % returns diagonal matrix D of eigenvalues and matrix
V whose columns are the corresponding right eigenvectors, so that
A*V = V*D.
eigValue=diag(D);
[IX]=sort(eigValue,'descend');
eigVector=V(:,IX);
```

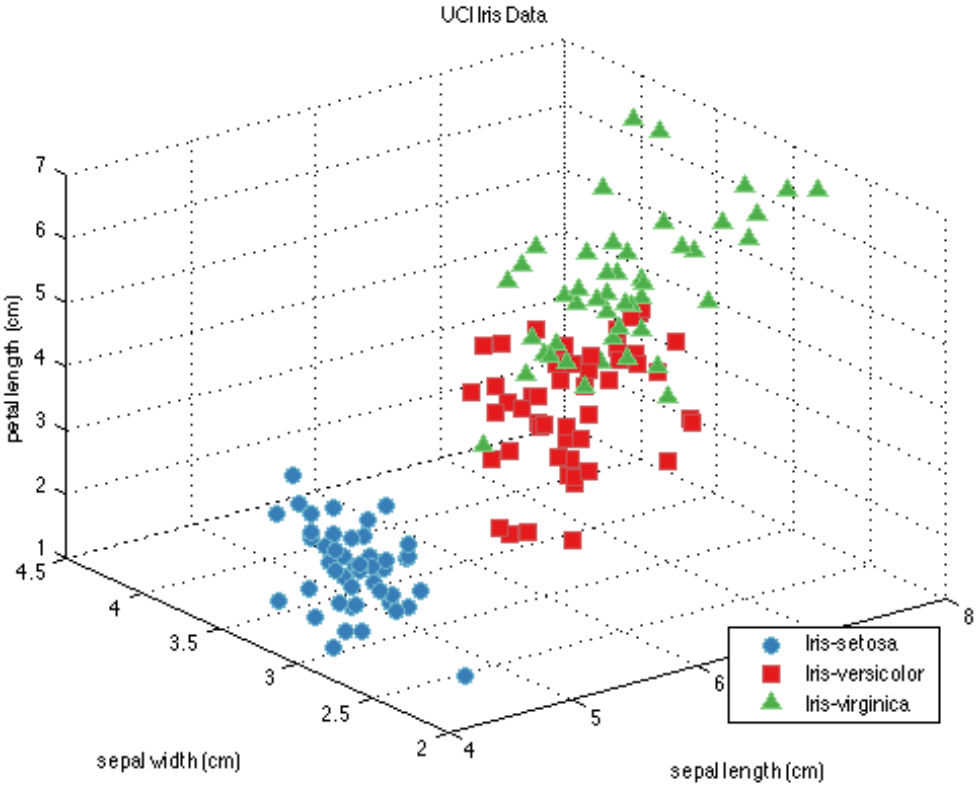
```
eigValue=eigValue(IX);

%% normailization
norm_eigVector=sqrt(sum(eigVector.^2));
eigVector=eigVector./repmat(norm_eigVector,size(eigVector,1),1);
%% dimensionality reduction
eigVector=eigVector(:,1:d);
Y=K0*eigVector;
```

7. Results and discussion:

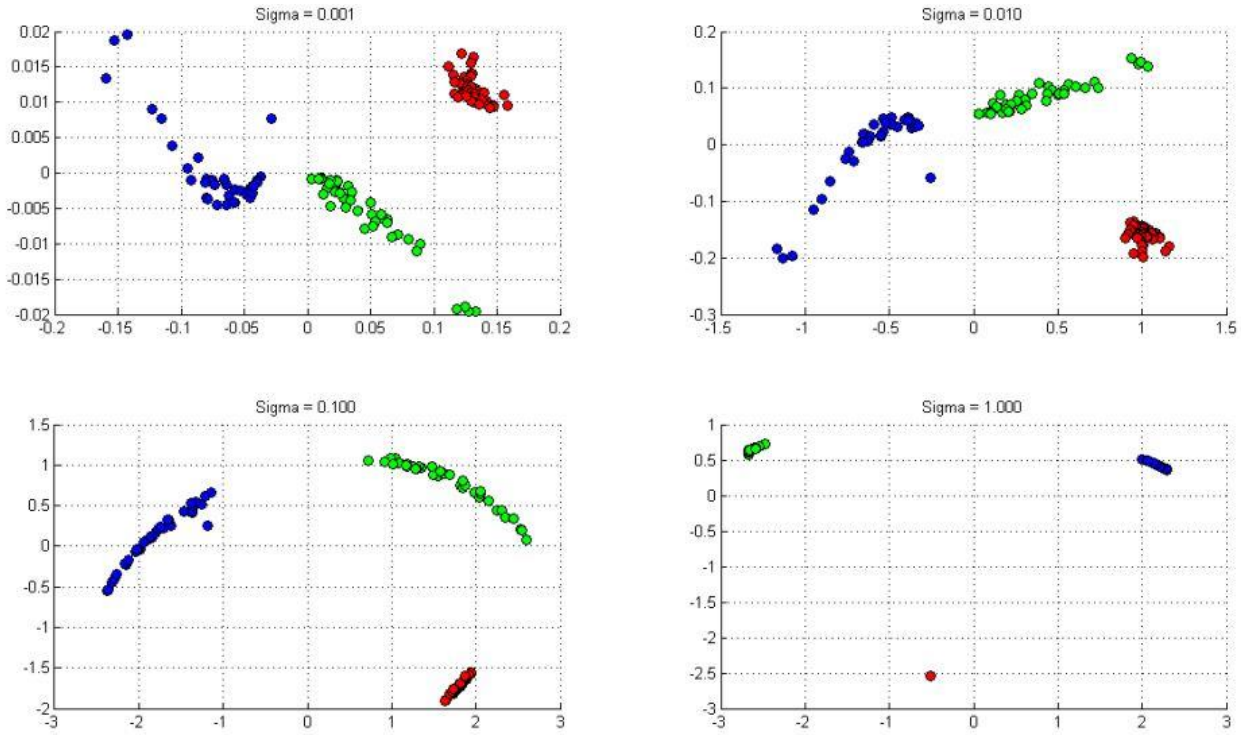
Original data in the original space

Color-code:

Iris-Virginica **Iris-Versicolor** **Iris-Setosa**



Iris dataset after performing Gaussian Kernel Method



We obtained the Iris dataset and we performed the Gaussian Kernel Method Trick to dimensionally reduce our data for different values of sigma and we found that the best result is achieved when sigma is equal to unity.

8. References

[1] Roman Rosipal, Mark Girolami, Leonard J. Trego and Andrzej Cichocki, Kernel PCA for Feature Extraction and De-noising in Non-linear Regression – 2001

[2] Daniel Olson, Applications and Implementation of Kernel Principal Component Analysis to Specific Data Sets - 2011.

[3] Rene Vidal, Yi Ma, S.Shankar Sastry, Generalized Principle Component Analysis - May 2006.