Assignment$_9$

Chaitra , Ganesh , Avik

July 12 2019

1 Question 1

In this problem we will compare the efficacy of PCA and k-PCA at separating datasets. For that we will use the two moon dataset. As you can see there are two classes there (0 and 1).

```
# coding: utf-8
```

```
# In[1]:
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.decomposition import KernelPCA
import math
```

```
# In[2]:
```

```
get_ipython().run_line_magic('matplotlib', 'inline')
```

```
# In[3]:
```

```
dataFrame = pd.read_csv('./data/Data_Sheet_9.csv')
features = dataFrame.iloc[1:,0:2]
features.shape
```

```
# In[4]:


pcaMoonDataSet = PCA(n_components=2)
principalComponentsMoonData = pcaMoonDataSet.fit_transform(features)
print(principalComponentsMoonData)


# In[5]:


plt.title("Moon dataset")
plt.xlabel("Feature1")
plt.ylabel("Feature2")
plt.scatter(features.iloc[1:298,0],features.iloc[1:298,1], color='red')
plt.scatter(features.iloc[299:,0],features.iloc[299:,1], color='blue')
plt.show()

plt.title("Plot on Linear-PCA")
plt.xlabel("PCA1")
plt.ylabel("PCA2")
plt.scatter(principalComponentsMoonData[1:298,0],principalComponentsMoonData[1:2
plt.scatter(principalComponentsMoonData[299:,0],principalComponentsMoonData[299:
plt.show()


# In[6]:


variance = 0.1
gammaForRBFKernel = 1/(2*math.pow(variance,2))
kernelPCA = KernelPCA(n_components=2,kernel='rbf',gamma=gammaForRBFKernel)
kernelPCAMoonDataSet = kernelPCA.fit_transform(features)
plt.title("Plot on Kernel-PCA")
plt.xlabel("PCA1")
plt.ylabel("PCA2")
plt.scatter(kernelPCAMoonDataSet[1:298,0],kernelPCAMoonDataSet[1:298,1], color='
plt.scatter(kernelPCAMoonDataSet[299:,0],kernelPCAMoonDataSet[299:,1], color='bl
plt.show()


# In[7]:


plt.title("Plot on first component Kernel-PCA")
plt.xlabel("PCA1")
```

```
plt.scatter(kernelPCAMoonDataSet[1:298,0],np.zeros(297), color='red')
plt.scatter(kernelPCAMoonDataSet[299:,0],np.zeros(299), color='blue')
plt.show()
```

a) Implement linear PCA on the dataset. Is the data when projected onto any of the principal components capable of separating the data by a linear separator? Plot the data projected on the linear components to see. The matplotlib library will be useful for this.

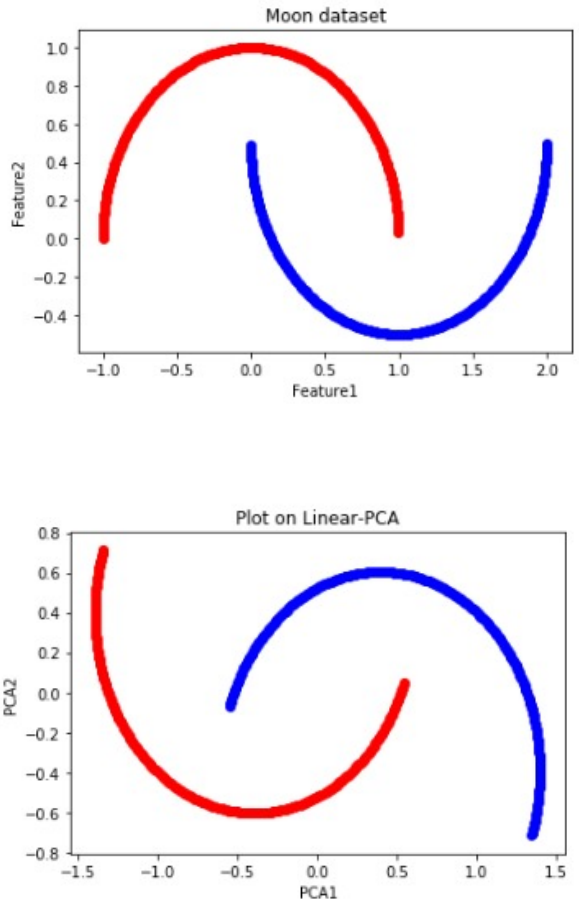

Figure 1: Plot on Linear PCA

Here we can clearly see that after projecting the data points on the PC-1 there is no clear separation of the data by a linear separator.

b) Implement and run kernel PCA on the dataset. Use the Gaussian kernel with $\sigma = 0.1$, you do not need to center the data. As in the previous problem look at the one dimensional projection of the data on the first few principal components, are any of them capable of linearly separating the data?

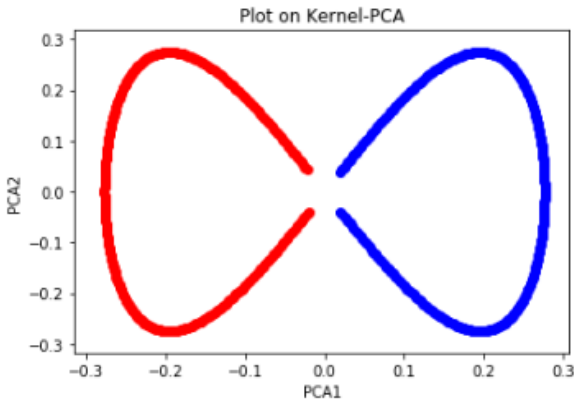

Figure 2: Plot on Kernel PCA

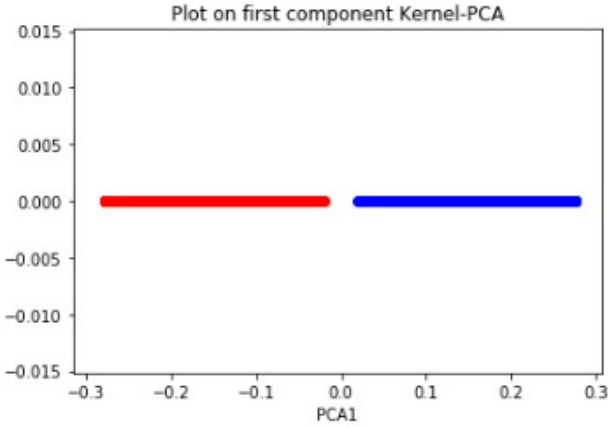

Figure 3: First principal component after Kernel PCA

After projecting the data on the PC-1 we can see clear distinction of the data by a linear separator.

2 Question 2

Prove the following theorem: The optimal kPCA solution $\alpha^* = (\alpha_1^*, ...\alpha_n^*)$ are the k largest Eigenvectors of the (centered) kernel matrix K.
Proof:
Let K be the centered kernel matrix.
We know that, (Lecture 9 Slide 35)

$$kPCA = max_{\alpha \in R^{nxk}} \sum_{j=1}^{k} \alpha_j^T \phi(X)^T \phi(X) \phi(X)^T \phi(X) \alpha_j$$

$$= max_{\alpha \in R^{nxk}} \sum_{j=1}^{k} \alpha_j^T K^2 \alpha_j \longrightarrow ①$$

subjecting to, $\alpha_j^T K \alpha_j = 1 \forall j$ Now by applying the Lagrangian multipliers, we get ① as

$$f(\alpha) = max_\alpha \alpha^T K^2 \alpha - \lambda \alpha^T \alpha$$

Differentiating the above equation w.r.t α,

$$\alpha \alpha^T = ||\alpha||^2$$

$$f'(\alpha) = 2K^2\alpha - 2\alpha\lambda = 0$$

$$f'(\alpha) = 2(K^2\alpha - \alpha\lambda) = 0$$

$$\alpha K^2 = \alpha\lambda$$

We can observe that the α^* is the eigenvector of K as we are trying to maximize the objective, α^* corresponds to the largest eigenvector of K. Hence proved.