LAB EXERCISE – 11

Principal Component Analysis

Aim of the Experiment

To write python program for finding principal component analysis (PCA) for the given dataset and to a randomly generated dataset.

Consider the dataset

$$\begin{pmatrix} 2 & 1 \\ 6 & 7 \end{pmatrix}$$

Apply PCA and Inverse Transform and Prove that they are similar.

In listing 2, the methods of computing mean matrix, covariance matrix, eigen values and eigen vectors computed are illustrated.

In listing 3, Iris dataset is taken and PCA is applied. It can be verified that after applying PCA, the cross score remains unchanged. That means, all the features of Iris are not important.

Listing 1

```
import numpy as np

from sklearn import decomposition

X = np.array([[2,6],[1,7]])


print("Orginal Matrx X and its Shape")

print(X)

print("Original Shape:",X.shape)

print("Original matrix\n\n")

# Apply Transform for X


pca = decomposition.PCA(n_components=2)

X_pca = pca.fit_transform(X)


print("Transformed Matrix and its Shape")
```

```
print(X_pca)
print("Transformed Shape:",X_pca.shape)
print("Transformed Matrix\n\n")

# Apply Inverse Transform

print("After Inverse Transform")
X_new=pca.inverse_transform(X_pca)
print(X_new)
print("After Inverse Transform\n\n\n")


# Explain variance
print('Explained variance\n')
print(pca.explained_variance_ratio_)
print('completed\n\n')

print('Singular values')
print(pca.singular_values_)
print('completed\n\n')
```

Output

```
Orginal Matrx X and its Shape
[[2 6]
 [1 7]]
Original Shape: (2, 2)
Original matrix


Transformed Matrix and its Shape
[[-7.07106781e-01  1.18606713e-17]
 [ 7.07106781e-01  1.18606713e-17]]
Transformed Shape: (2, 2)
Transformed Matrix


After Inverse Transform
[[2. 6.]
 [1. 7.]]
After Inverse Transform



Explained variance

[1.00000000e+00 2.81351049e-34]
completed


Singular values
[1.00000000e+00 1.67735223e-17]
completed
```

<u>Listing 2</u>

<u>This explains how the eigen values and eigen vectors are calculated.</u>

```
import numpy as np

from numpy.linalg import eig


# define a matrix

X = np.array([[3, 6], [4,7]])


print("Orginal Matrx X and its Shape")

print(X)


print("Original matrix Shape")

print("Original Shape:",X.shape)
```

```
# calculate the mean of each column
M = np.mean(X.T, axis=1)
print("\nMean matrix")
print(M)



# center columns by subtracting column means
C = X - M
print("\nCentre the matrix")
print(C)

# calculate covariance matrix of centered matrix
V = np.cov(C.T)
print("\nCovariance of the matrix\n")
print(V)

# eigendecomposition of covariance matrix
values, vectors = eig(V)
print('\n Eigen vectors')
print(vectors)

print('\n Eigen values')
print(values)
```

Output

```
In [10]: runfile('D:/Test/lab6-detailedpca.py', wdir='D:/Test')
Orginal Matrx X and its Shape
[[3 6]
 [4 7]]
Original matrix Shape
Original Shape: (2, 2)

Mean matrix
[3.5 6.5]

Centre the matrix
[[-0.5 -0.5]
 [ 0.5  0.5]]

Covariance of the matrix

[[0.5 0.5]
 [0.5 0.5]]

 Eigen vectors
[[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]

 Eigen values
[1.00000000e+00 1.11022302e-16]
```

Listing 3

```
import pandas as pd

import numpy as np

from sklearn.model_selection import Kfold

from sklearn import preprocessing

from sklearn.model_selection import train_test_split

from sklearn.model_selection import cross_val_score

from sklearn.neighbors import KneighborsClassifier

#from sklearn.naive_bayes import GaussianNB

from sklearn import decomposition

import seaborn as sns



df = pd.read_csv("iris.csv")

print(df.head(10))

array = df.values
```

```
X = array[:,0:4]
y = array[:,4]

kfold = KFold(n_splits=10)
model = KneighborsClassifier(n_neighbors=3)
#model = GaussianNB()
#or use any other classifier of your choice
score = cross_val_score(model,X,y,cv=10)
print('\n\n')
print("Cross score before applying PCA\n")
print(score.mean())

print("Apply PCA now...")

pca = decomposition.PCA(n_components=1)
X_pca = pca.fit_transform(X)
core = cross_val_score(model,X_pca,y,cv=10)
print('\n\n')
print("Cross score After applying PCA\n")
print(score.mean())
```

Output

```
In [21]: runfile('D:/Test/Lab6A-PCA-Iris.py', wdir='D:/Test')
   sepal.length  sepal.width  petal.length  petal.width variety
0           5.1          3.5           1.4          0.2  Setosa
1           4.9          3.0           1.4          0.2  Setosa
2           4.7          3.2           1.3          0.2  Setosa
3           4.6          3.1           1.5          0.2  Setosa
4           5.0          3.6           1.4          0.2  Setosa
5           5.4          3.9           1.7          0.4  Setosa
6           4.6          3.4           1.4          0.3  Setosa
7           5.0          3.4           1.5          0.2  Setosa
8           4.4          2.9           1.4          0.2  Setosa
9           4.9          3.1           1.5          0.1  Setosa



Cross score before applying PCA

0.9666666666666666
Apply PCA now...



Cross score After applying PCA

0.9666666666666666
```

If more attributes are removed, it leads to more information loss. So optimal replacement of features is required. In the above case, retaining only one component does not result in reduction of scores.