Practical - PCA

- Shivam Tawari (A-58)

Aim: Write a python program to evaluate to apply PCA algorithm.

Theory:

It is a statistical technique to convert high dimensional data to low dimensional data by selecting the most important features that capture maximum information about the dataset.

The features are selected on the basis of variance that they cause in output. The feature that causes highest variance is first principal component.

The feature that is responsible for second highest variance is considered the second Principal component, and so on.

Advantages:

1. Removes Correlated Features
2. Improves Algorithm Performance
3. Reduces Overfitting
4. Improves visualization

Disadvantages:

① Independent variables become less interpretable
② Data standardization is must before PCA.
③ Information Loss.

Code:

```
from numpy import array
from numpy import mean
from numpy import cov
from numpy.linalg import eig.

A = array([[1,2], [3,4], [5,6]])
print (A)

M = mean(A.T, axis = 1)
print (M)

C = A - M
print (C)

V = cov(C.T)
print (V)

values, vectors = eig (V)
print (values)
```

```
print (vectors)
P = vectors.T.dot(C.T)
print (P.T)
```

Example 2 :

```
import numpy as np
import pandas as pd

data = "iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length',
         'petal-width', 'class']
dataset = pd.read_csv(data, names = names)

dataset.head()

x = dataset.drop('class', 1)
y = dataset['class']

from sklearn.model_selection import train-test-split
x_train, x_test, y_train, y_test = train_test_split
                                   (x, y, test_size = 0.2,
                                    random_state = 0)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
```

```
X_test = sc.transform (X_test)

from sklearn.decomposition import PCA
pca = PCA()
X_train = pca.fit_transform (X_train)
X_test = pca.transform (X_test)

explained_variance = pca.explained_
                     variance_ratio_
print (explained_variance
```

Conclusion:
Hence, we have successfully implemented the python program for principal component analysis.

Code:

Example 1:

```
[1]  # Shivam Tawari A-58
     from numpy import array
     from numpy import mean
     from numpy import cov
     from numpy.linalg import eig
```

```
[2]  A = array([[1, 2], [3, 4], [5, 6]])
     print(A)
```

```
     [[1 2]
      [3 4]
      [5 6]]
```

```
[3]  M = mean(A.T, axis=1)
     print(M)
```

```
     [3. 4.]
```

```
[4]  C = A - M
     print(C)
```

```
     [[-2. -2.]
      [ 0.  0.]
      [ 2.  2.]]
```

```
[5]  V = cov(C.T)
     print(V)
```

```
     [[4. 4.]
      [4. 4.]]
```

```
[6]  values, vectors = eig(V)
     print(values)
     print(vectors)
```

```
     [8. 0.]
     [[ 0.70710678 -0.70710678]
      [ 0.70710678  0.70710678]]
```

```
[7]  P = vectors.T.dot(C.T)
     print(P.T)
```

```
     [[-2.82842712  0.        ]
      [ 0.          0.        ]
      [ 2.82842712  0.        ]]
```

Example 2:

```
[12] # Shivam Tawari A-58
     import numpy as np
     import pandas as pd

     data="iris.data"
     names=['sepal-length','sepal-width','petal-length','petal-width','Class']
     dataset=pd.read_csv(data,names=names)

     dataset.head()

     X=dataset.drop('Class',1)
     y=dataset['Class']

     from sklearn.model_selection import train_test_split
     X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)

     from sklearn.preprocessing import StandardScaler

     sc=StandardScaler()
     X_train=sc.fit_transform(X_train)
     X_test=sc.transform(X_test)

     from sklearn.decomposition import PCA
     pca=PCA()
     X_train=pca.fit_transform(X_train)
     X_test=pca.transform(X_test)
```

```
explained_variance=pca.explained_variance_ratio_
print(explained_variance)
```

```
[0.72226528 0.23974795 0.03338117 0.0046056 ]
```