Practical-12

Objective:

By using PCA perform dimensionality reduction .

Description

Given a 2D data set perform dimensionality reduction by using PCA technique and convert it to 1D data points and test the difference.

Dataset:

1. Name: Social_Network_Ads.
2. Source: https://www.kaggle.com/dragonheir/logistic-regression
3. This is a basic dataset designed for learning logistic regression.
4. It shows whether a person(Male or Female) with salary(s) has bought a product or not 5.
5. It has 400 samples.

Procedure:

1. Read data from csv using pandas
2. Consider only female records
3. Drop ID and gender
4. Covert dataset into list of x(data) and y(target)
5. Split the data into training and testing data
6. Perform logistic regression and note down accuracy
7. PCA
    - Compute the mean along feature
    - Compute center matrix by subtracting x by it's feature mean respectively
    - Compute co-variance matrix on center
    - Compute Eigen values and vectors and normalize it(consider largest value for calculating Eigen vector)
    - Perform dot product of transpose(considered Eigen vector) and data samples to get reduced 1D data
8. Perform PCA on x and reshape it
9. Perform logistic regression and note down accuracy

Implementation:

```
# 1 read data
import pandas as pd
data = pd.read_csv('./Social_Network_Ads.csv')
```

```
# 2 Consider only female records
data = data[data['Gender']=='Female']

# 3 Drop gender and ID
data = data.drop(columns=['Gender','User ID'])

# 4 Covert dataset into list of x(data) and y(target)
x = data.iloc[:,:-1].values
y = data.iloc[:,-1].values

# 5 split data into test and training
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30)

# 6 Logistic reg
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
from sklearn.metrics import accuracy_score
print("accuracy before : ", accuracy_score(y_test, y_pred))

# 7 PCA
import numpy as np
def PCA(data):
 mean = np.mean(data.T, axis=1)
 print("MEAN : ")
 print(mean)
 center = data - mean
 co_var = np.cov(center.T)
 print("Co-variance : ")
 print(co_var)
 e_values, e_vectors = np.linalg.eig(co_var)
 print("Eigen values : ")
 print(e_values)
 print("Eigen vectors : ")
 print(e_vectors)
 considered_e_values = np.array(e_vectors[0])
 print("Considered Eigen vector : ")
 print(considered_e_values)
 reduced = list()
 for i in center:
 reduced.append(considered_e_values.T.dot(i))
 return reduced

# 8 Preform PCA on X and reshape it
x = data.iloc[:,:-1].values
x = PCA(x)
```

```
x = np.array(x).reshape(-1, 1)
y = data.iloc[:,-1].values

# 9 Perform logistic regression and note down accuracy
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.3)
model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print("accuracy after : ", accuracy_score(y_test, y_pred))
```

Output:

accuracy before : 0.6129032258064516
accuracy after : 0.8225806451612904

MEAN : [3.84117647e+01 7.17598039e+04]

Co-variance : [[1.17918285e+02 6.78530861e+04] [6.78530861e+04 1.26702084e+09]]

Eigen values : [1.14284531e+02 1.26702084e+09]

Eigen vectors : [[-9.99999999e-01 -5.35532566e-05] [5.35532566e-05 -9.99999999e-01]]

Considered Eigen vector : [-9.99999999e-01 -5.35532566e-05]