question4

April 12, 2023

[3]:

```
# Question 4
# Anirudh Sathish
# CS20B1125
```

0.0.1 4. Eigenfaces-Face classification using PCA (40 classes)

- a) Use the following “face.csv” file to classify the faces of 40 different people using PCA.
- b) Do not use the in-built function for implementing PCA.
- c) Use appropriate classifier taught in class (use any classification algorithm taught in class like Bayes classifier, minimum distance classifier, and so on)

[4]:

```
# libs
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
/home/anirudh/.local/lib/python3.8/site-
packages/pandas/core/computation/expressions.py:20: UserWarning: Pandas requires
version '2.7.3' or newer of 'numexpr' (version '2.7.1' currently installed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED
```

[5]:

```
# get data
faces = pd.read_csv("face.csv")
faces.head()
```

[5]:

```
          0         1         2         3         4         5         6  \
0  0.309917  0.367769  0.417355  0.442149  0.528926  0.607438  0.657025
1  0.454545  0.471074  0.512397  0.557851  0.595041  0.640496  0.681818
2  0.318182  0.400826  0.491736  0.528926  0.586777  0.657025  0.681818
3  0.198347  0.194215  0.194215  0.194215  0.190083  0.190083  0.243802
4  0.500000  0.545455  0.582645  0.623967  0.648760  0.690083  0.694215

          7         8         9  …      4087      4088      4089      4090  \
0  0.677686  0.690083  0.685950  …  0.669422  0.652893  0.661157  0.475207
1  0.702479  0.710744  0.702479  …  0.157025  0.136364  0.148760  0.152893
2  0.685950  0.702479  0.698347  …  0.132231  0.181818  0.136364  0.128099
3  0.404959  0.483471  0.516529  …  0.636364  0.657025  0.685950  0.727273
```

```
4  0.714876  0.723140  0.731405  …  0.161157  0.177686  0.173554  0.177686

       4091      4092      4093      4094      4095  target
0  0.132231  0.148760  0.152893  0.161157  0.157025       0
1  0.152893  0.152893  0.152893  0.152893  0.152893       0
2  0.148760  0.144628  0.140496  0.148760  0.152893       0
3  0.743802  0.764463  0.752066  0.752066  0.739669       0
4  0.177686  0.177686  0.177686  0.173554  0.173554       0

[5 rows x 4097 columns]
```

Observation

- There are around 4097 columns here , let us reduce this to around 4

[6]:

```
# seperating the target and the features
y = faces["target"]
X = faces.iloc[:,:-1]
```

[7]:

```
# implement function for PCA
def PCA(df , k):
    X = df.to_numpy()

    # standardise
    X_std = (X - np.mean(X, axis=0)) / np.std(X, axis=0)

    cov_matrix = np.cov(X_std.T)
    eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)
    eig_pairs = [(np.abs(eigenvalues[i]), eigenvectors[:,i]) for i in␣
 ↪range(len(eigenvalues))]
    eig_pairs.sort(reverse=True, key=lambda x: x[0])
    matrix_w = np.hstack((eig_pairs[i][1].reshape(len(df.columns),1)) for i in␣
 ↪range(k))
    X_pca = X_std.dot(matrix_w)

    # Convert the transformed data back to a DataFrame
    cols = [f"PC{i+1}" for i in range(k)]
    df_pca = pd.DataFrame(X_pca, columns=cols)

    return df_pca
```

[8]:

```
dim = 32
X_ = PCA(X,dim)
```

```
/tmp/ipykernel_25105/916561548.py:12: FutureWarning: arrays to stack must be
passed as a "sequence" type such as list or tuple. Support for non-sequence
iterables such as generators is deprecated as of NumPy 1.16 and will raise an
```

```
error in the future.
  matrix_w = np.hstack((eig_pairs[i][1].reshape(len(df.columns),1)) for i in
range(k))
```

[9]:

```
X_reduced = pd.DataFrame(np.real(X_))
```

[10]:

```
# let's split into test and train
from sklearn.model_selection import train_test_split

# random state for reproducability
X_train, X_test , y_train , y_test = train_test_split(X_reduced,y,test_size=0.
 ↪3,random_state= 2)
```

0.0.2 Further

Now we need to train this using a classifier

[11]:

```
# lets use the minimum dist classifier
def dist(X,mean):
    Y = np.sqrt(np.sum((X - mean)**2))
    return Y
```

[12]:

```
target = pd.DataFrame(y_train.to_numpy())
target.columns = ["label"]
X_recomplete = pd.concat([pd.DataFrame(X_train.to_numpy()),target],axis = 1)
X_recomplete
```

[12]:

```
             0          1          2          3          4          5  \
0   -12.470149 -22.764214  25.854409  -7.872594 -29.778225  15.559930
1   -13.452591 -38.498148   6.261032   2.939254   4.481049   4.556061
2   -35.465647  -4.926576  -7.895512  13.953842  -5.543100  -6.879875
3   -25.776232 -24.063425  10.668826   1.931122  -3.718425  -2.880087
4     0.238321   8.657132  18.254694  13.704750  -2.343360 -21.417495
..         …          …          …          …          …          …
275 -17.880372   5.887619   4.140011  21.170258  -5.287500 -10.634399
276 -13.796903  24.401468   7.964249   1.970064   5.356964   3.561706
277  48.136781  30.840925  -1.333773   9.199799  20.946557 -14.491309
278  -1.565518  26.154681 -28.312928  -2.682017  14.476955   0.294051
279  18.489601 -48.362660   5.913987 -21.389318  -8.503165  -8.524571

             6          7          8          9  …        23        24  \
0     4.050746   8.460022  -0.199687 -11.354322  … -1.081742 -3.656365
1     5.891295  -0.457788  -5.253466  -6.895147  … -3.228498 -3.974198
2     3.305456  -4.281920  -3.479988 -10.934135  …  0.999866 -2.149057
3    21.570968  10.836741   1.605833   4.938735  … -0.767468 -0.786237
4    -0.625946 -12.444465   5.960021  -0.003394  … -0.777802  0.930801
..         …          …          …          …  …  …        …
```

```
275   3.866715   0.174368  -3.719179   2.081378  …  1.677615  0.409395
276   1.629407 -20.914326  -5.768807  -5.083873  … -2.919046  1.519434
277 -24.954809   3.914137   0.620176  -2.174642  … -0.562042  7.657260
278  18.801110  11.354332  -5.577051   5.767145  … -1.274335 -7.426593
279  -5.861535   1.067542  10.100787  11.317871  … -2.861158 -4.906727

           25        26         27        28        29        30        31  \
0    0.817381 -4.961834   6.313481 -5.174302 -1.528832  0.941685  0.669370
1    0.279343 -2.051225  -4.910065 -0.264720 -0.686249  1.245296  4.396015
2    1.761309  1.730883  -2.350602 -0.423686 -7.856503 -1.560900  0.591192
3    6.978522  7.386602  -5.555768 -2.648823 -0.337393 -3.605931 -6.937106
4    0.050183  1.712293   9.910267 -5.100609  1.578396 -3.644937  1.209926
..        …        …         …       …       …       …       …
275 -4.457368  2.471959   5.820890  1.041556 -0.472682 -0.116670  0.312237
276 -0.648941 -1.746017  -3.288061 -3.427770  2.763512  1.830751 -5.572531
277  6.196506  1.104143  11.503930  1.715735 -2.183316 -1.218641  1.807812
278  3.318042  1.828388   0.512862  1.052292 -9.496571 -4.267750 -4.922249
279  7.471773 -1.812927  -9.581378  3.887263 -6.112964  6.345869  1.775716

     label
0       11
1       20
2       29
3       30
4       34
..     …
275     29
276      2
277      7
278      1
279     16

[280 rows x 33 columns]
```

[13]:

```
meanData = X_recomplete.groupby("label").mean()
```

[14]:

```
predictions = []
for i in range(X_test.shape[0]):
    min = 10000000
    minIndex = -1
    for w in range(len(meanData)):
        dist_t = dist(X_test.iloc[i:i+1].to_numpy(),meanData.xs(w).to_numpy())
        if dist_t < min:
            min = dist_t
            minIndex = w
    predictions.append(minIndex)
```

[15]:

```
y_test = pd.DataFrame(y_test)
```

[16]:

```
# checking for accuracy
match = 0
for i in range(len(y_test)):
    expected = y_test["target"].iloc[i]
    prediction = predictions[i]
    if(prediction == expected):
        match+=1

accuracy = (match/len(y_test))*100

print("Accuracy of Classifier :" ,accuracy)
```

```
Accuracy of Classifier : 85.83333333333333
```