Project Report 1

Name : DHIVAKAR.R

Course : AI and ML

Problem Statement : Face Feature Extraction Using PCA

Prerequisites :

The needed software were :

Python 3.6 This setup requires that your machine has latest version of python. The following url https://www.python.org/downloads/ can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables , if you want to run python program directly, detail instructions are below in how to run software section. To do that check this: https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external- command/. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic.

Second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url https://www.anaconda.com/download/ You will also need to download and install below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run below commands in command prompt/terminal to install these packages pip install -U scikit-learn pip install numpy pip install scipy if you have chosen to install anaconda then run below commands in anaconda prompt to install these packages conda install -c scikit-learn conda install -c anaconda numpy conda install -c anaconda scipy .

Dataset Used :

The Data source used for this projet is been generated using sklearn library .

Method used for Detection

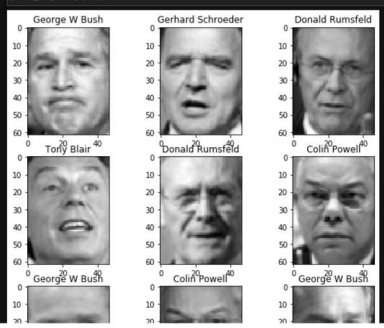
PCA

Importing the libraries and capturing images :

```
[1]: from sklearn.datasets import fetch_lfw_people
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import classification_report
     from sklearn.decomposition import PCA
     from sklearn.neural_network import MLPClassifier
     import numpy as np
     import matplotlib.pyplot as plt

[2]: # Preparing Dataset
     dataset = fetch_lfw_people(min_faces_per_person=100, resize=0.5)
     X = dataset.data
     y = dataset.target
     target_names = dataset.target_names
     images = dataset.images

     Downloading LFW metadata: https://ndownloader.figshare.com/files/5976012
     Downloading LFW metadata: https://ndownloader.figshare.com/files/5976009
     Downloading LFW metadata: https://ndownloader.figshare.com/files/5976006
     Downloading LFW data (~200MB): https://ndownloader.figshare.com/files/5976015
```

Plot the images

```
[3]: X.shape

[3]: (1140, 2914)

[4]: n, h, w = images.shape
     print(len(target_names))
     np.unique(y, return_counts = True)

     5
[4]: (array([0, 1, 2, 3, 4], dtype=int64),
      array([236, 121, 530, 109, 144], dtype=int64))

[5]: print(n)
     print(h)
     print(w)

     1140
     62
     47

[6]: # Plot the images
     def plot_img(images, titles, h, w, rows=3, cols=3):
         plt.figure(figsize=(3*cols, 3*rows))
         for i in range(rows*cols):
             plt.subplot(rows, cols, i+1)
             plt.imshow(images[i].reshape(h,w), cmap="gray")
             plt.title(target_names[titles[i]])
     plot_img(X, y, h, w)
```



Train the data

```
[7]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1)

[8]: X_train.shape

[8]: (1026, 2914)

[9]: pca = PCA()
     pca.fit(X_train)

[9]: PCA(copy=True, iterated_power='auto', n_components=None, random_state=None,
         svd_solver='auto', tol=0.0, whiten=False)

[10]: pca.transform(X_train).shape

[10]: (1026, 1026)

[11]: var = pca.explained_variance_
      print(var)
      comp = pca.components_
      print(comp.shape)

      [7.3633869e+05 6.3650162e+05 3.0940647e+05 ... 3.2197502e+00 2.9875937e+00
       6.8564386e-06]
      (1026, 2914)

[12]: val_sum = np.sum(var)
      print(val_sum)
      sort_indx = np.argsort(var)
      sort_indx = sort_indx[::-1]
      print(sort_indx)

      4145777.8
      [   0    1    2 ... 1023 1024 1025]

[13]: temp_sum = 0
      principal_vec = []
      principal_val = []
      i = 0
      while (temp_sum < 0.98*val_sum):
          principal_vec.append(comp[sort_indx[i], :])
          principal_val.append(var[sort_indx[i]])
          temp_sum += var[sort_indx[i]]
          i += 1
      print("Number of components : {}".format(i))

      Number of components : 272

[14]: principal_vec = np.matrix(principal_vec)
      print(principal_vec.shape)

      (272, 2914)

[15]: X_train_transf = np.dot(X_train, principal_vec.T)
      X_test_transf = np.dot(X_test, principal_vec.T)

[16]: X_train_transf.shape
```

```
[16]: X_train_transf.shape

[16]: (1026, 272)

[17]: clf = MLPClassifier(hidden_layer_sizes = (1024, ), batch_size = 128, verbose = True, early_stopping = True)
      clf.fit(X_train_transf, y_train)

      Iteration 1, loss = inf
      Validation score: 0.543689
      Iteration 2, loss = inf
      Validation score: 0.514563
      Iteration 3, loss = 56.89804045
      Validation score: 0.689320
      Iteration 4, loss = 19.15569836
      Validation score: 0.786408
      Iteration 5, loss = 11.50287817
      Validation score: 0.805825
      Iteration 6, loss = 4.97681593
      Validation score: 0.776699
      Iteration 7, loss = 3.38226608
      Validation score: 0.873786
      Iteration 8, loss = 2.15541016
      Validation score: 0.883495
      Iteration 9, loss = 0.59051996
      Validation score: 0.844660
      Iteration 10, loss = 0.19031752
      Validation score: 0.883495
      Iteration 11, loss = 0.07437655
      Validation score: 0.883495
      Iteration 12, loss = 0.04975753
      Validation score: 0.883495
      Iteration 13, loss = 0.00043571
      Validation score: 0.883495
      Iteration 14, loss = 0.00074897
      Validation score: 0.873786
      Iteration 15, loss = 0.00016122
      Validation score: 0.873786
      Iteration 16, loss = 0.00015903
```

```
                validation_fraction=0.1, verbose=True, warm_start=False)

[18]: y_pred = clf.predict(X_test_transf)
      print(classification_report(y_test, y_pred, target_names = target_names))

                     precision    recall  f1-score   support

          Colin Powell       0.75      0.86      0.80        21
       Donald Rumsfeld       0.75      0.75      0.75        12
         George W Bush       0.88      0.87      0.88        61
     Gerhard Schroeder       0.71      0.83      0.77         6
            Tony Blair       0.91      0.71      0.80        14

              accuracy                           0.83       114
             macro avg       0.80      0.80      0.80       114
          weighted avg       0.84      0.83      0.83       114
```

Plot the images after applying pca

```
[20]: # Plot the images after applying pca
def plot_img(images, titles, h, w, rows=3, cols=3):
    plt.figure(figsize=(4*cols, 4*rows))
    for i in range(rows*cols):
        plt.subplot(rows, cols, i+1)
        plt.imshow(images[i].reshape(h,w), cmap="gray")
        plt.title([titles[i]])
n_components = 272
mean_imgs = []
for i in range(n_components):
    vec = principal_vec[i,:]
    img = vec.reshape((h, w))
    mean_imgs.append(img)
mean_imgs = np.array(mean_imgs)
print(mean_imgs.shape)

(272, 62, 47)
```

Output

```
[21]: pca_titles = [f"eigenvector {i}" for i in range(n_components)]
plot_img(mean_imgs, pca_titles, h, w)
```