Anas Emad 7048

Mazen Ahmed Ghanem 6896

Mohab Ayman 7127

Assignment 1 Pr

Face Recognition using LDA &PCA

**2. Generate the Data Matrix and the Label vector**

**Text

Description automatically generated**

We have a for loop to get the list of files in our directory which will be from s1 to s40 each having 10 images for 1 of the 40 persons In the dataset.

For each image in the directory, the code uses the cv2.imread() function from the OpenCV library to read the image as a NumPy array.

We then flatten the images and append them to our data matrix and we take the label which is a number from 1 to 40 and we append it to our label vector

**Split the Dataset into Training and Test sets**

**Text

Description automatically generated**

We split the data and labels of each person to have 5 images in the test dataset and 5 in the training dataset by splitting them into even and odd images and adding each image to the respective dataset by using the above for loop

Text

Description automatically generated

**4. Classification using PCA**

**Text

Description automatically generated**

Text

Description automatically generated

We have a function pca(X, alpha): which takes X the data matrix and alpha :the explained variance. we calculate the mean for our data

Then we center the data by subtracting the mean from it.The covariance matrix is computed using **C = np.cov(X\_centered.T)**

The eigen values and eigen vectors are calculated using eigenvalues, **eigenvectors = np.linalg.eigh(C)**

We sort the eigen values in descending order to know the eigen vectors with maximal variance in data and we capture their indexes.

We then sort eigen vectors according to the indexes

**idx = np.absolute(eigenvalues).argsort()[::-1]**

**eigenvectors = eigenvectors[:,idx]**

we create a for loop to take enough eigen values to be equal to or surpass alpha

we store the number of components in n\_components .

**U = eigenvectors[:, :n\_components]**

**print("dim",X\_centered.shape,U.shape)**

**X\_reduced = np.matmul(X,U)**

**return X\_reduced, U**

we take the first n\_components eigen vectors to create our projection matrix we project the data by multiplying the data matrix with the projection matrix and the function returns the projected data and the projection matrix

**5. Classification Using LDA**

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

We have the function **lda\_org(data\_matrix,train\_label):** which takes the data matrix and the test data we calculate the total mean .we then calculate the mean for each class(5 rows) .and we center the data for each class by subtracting its man from the original data.

We calculate the class scatter matrices for each class and add them together in S\_total.

We calculate Sb by the equation given above.

We use 39 eigen vectors for the projection matrix and we calculate the projected data and we return the projected data and the projection matrix

Text

Description automatically generated

Lda gave us an accuracy of 0.955 which is slightly better than pca

6) classifier Tuning:

a)After running PCA,LDA algorithm we get the projected trained data and project the test data using the same projection matrix ,then use K-NN classifier with values

[i= 1,3,5,7] to get the k with best accuracy.

knn\_pca = KNeighborsClassifier(n\_neighbors=i)

b)there are multiple opinions in this part as in the scikit-learn the function chooses the class with the largest num of samples

another opinion is that the function chooses it randomly

and the third opinion is to choose the least weight of the distances.

c)PLOTS:

1)PCA:

Chart, line chart

Description automatically generated

2)LDA:

Chart, line chart

Description automatically generated

7)compare non-faces and faces:

a)first we loaded a dataset contains photos of flowers, dogs and cars ,then converted them to grayscale and resized the images to 92x112 then made the same operations as the original faces dataset to flatten the images and stack them into 1 data matrix and assign labels for the samples but in this case we have only 2 classes as it is a binary classification problem either face or non face .

next we split the data into train and test each will be 200\*10304 matrix.

i) we project the test and train data then get the confusion the matrix to get the number of failure and success cases after the prediction of the test data using K-NN using k=1.

Text

Description automatically generated

ii) the number of eigen vectors used in the projection matrix is equal to the number of [classes - 1] so in our case the number of classes is 2 and the number of the eigen vectors is 1 keeping in mind in case of increasing the number of eigen vectors the accuracy will increase.

iii)Plot fixed num of faces and variable number of non faces:

here we change the number of the nonfaces in test projection starting from 50 and every iteration we add more 30 non faces.

Text

Description automatically generated

Chart, line chart

Description automatically generated

iv)

the accuracy of LDA in case of large number of non faces in training set 50 samples of faces and 200 samples of nonfaces:

Text

Description automatically generated

Non\_faces accuracy:

Text

Description automatically generated

If you increased the number of non faces images in the training sample, the model will overfit or prefer to classify the data as non faces due to the large number of nonfaces and the small number faces data therefore this will cause that the faces can be classified as non faces too.

The accuracy of the LDA in case of large number of non faces in training set is higher than the nonfaces equal to faces.

Bonus Part:

a)splitting data into train and test with 70:30

the only change is in the original PCA and LDA we split data according to the index even or odd here we use [mod 10] to decide either the sample index is from 0->2 or from 3->9 to split it as 70% train and 30% test.

Accuracy comparison:

PCA: after new split

Text

Description automatically generated

LDA:

Graphical user interface, text

Description automatically generated with medium confidence

b)variations of PCA and LDA:

There are many variations of LDA and PCA implementation, one of them is in sklearn library (**from** **sklearn.discriminant\_analysis** **import** [LinearDiscriminantAnalysis](https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html#sklearn.discriminant_analysis.LinearDiscriminantAnalysis), **from** **sklearn.decomposition** **import** [PCA](https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA)).We used it and calculated the accuracy, confusion matrix, and time complexity of each approach to compare between them.

Text

Description automatically generated