Pattern Recognition
Assignment (1)
Face Recognition

Nancy Hossam ID: 5359

Hania Fayed ID: 5757

Hana Magdy ID: 5455

Fagr hesham ID: 5886

This is a face recognition assignment using 2 approaches: LDA & PCA.

Step one: Understanding the dataset and preparing it:

We worked on ORL dataset that consists of forty classes each consists of ten pictures to a specific person. Each picture captures a different facial expression for that person. Pictures are all grayscale with dimensions (92 x 112). Each picture is converted into one vector of dimensions (1 x 10304) each column represents an attribute of that picture. Then all vectors are stacked under each other in a total data matrix with dimensions (400x10304).

The Data set was separated into two sets testing and training. Each set includes two hundred pictures, five pictures for each person.

Their dimensions are (200x10304).

Step two: applying PCA algorithm:

ALGORITHM 7.1. Principal Component Analysis

PCA $(\mathbf{D}, \alpha)$:

1 $\mu = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$ // compute mean
2 $\mathbf{Z} = \mathbf{D} - \mathbf{1}\cdot\mu^T$ // center the data
3 $\mathbf{\Sigma} = \frac{1}{n}(\mathbf{Z}^T\mathbf{Z})$ // compute covariance matrix
4 $(\lambda_1, \lambda_2, \ldots, \lambda_d) = \text{eigenvalues}(\mathbf{\Sigma})$ // compute eigenvalues
5 $\mathbf{U} = (\mathbf{u}_1 \quad \mathbf{u}_2 \quad \cdots \quad \mathbf{u}_d) = \text{eigenvectors}(\mathbf{\Sigma})$ // compute eigenvectors
6 $f(r) = \frac{\sum_{i=1}^{r}\lambda_i}{\sum_{i=1}^{d}\lambda_i}$, for all $r = 1, 2, \ldots, d$ // fraction of total variance
7 Choose smallest r so that $f(r) \geq \alpha$ // choose dimensionality
8 $\mathbf{U}_r = (\mathbf{u}_1 \quad \mathbf{u}_2 \quad \cdots \quad \mathbf{u}_r)$ // reduced basis
9 $\mathbf{A} = \{\mathbf{a}_i \mid \mathbf{a}_i = \mathbf{U}_r^T\mathbf{x}_i, \text{for } i = 1, \ldots, n\}$ // reduced dimensionality data

The above figure shows PCA algorithms steps.

Further PCA explanation:

The PCA function takes two parameters the training set and the alpha.

- First, the mean is computed for each attribute in the training set
- We subtract the mean from the training set to center the data.
- The centered data is the used to compute the covariance matrix.
- The Eigen values and Eigen vectors are obtained from the covariance matrix.
- After computing the Eigen values we determine the number of dominant Eigen vectors will be taken into consideration by determining the fractional variance that satisfy the alpha.
- Finally the reduced Eigen vectors (projection matrix) are multiplied by the centered testing data set and training data set producing projected training data and projected testing data.
- We used KNN classifier to classify the projected testing set. KNN classifier determine the label for each element in testing set through calculating Euclidean distance it stores all available cases and takes into consideration the k smallest Euclidean distance to it.
  (in our case we used k=1"first nearest neighbor")

By applying the PCA on different values of alpha w observed that accuracy increases as alpha increases (number of eigenvectors taken into consideration increases) to a certain value of alpha then the accuracy decreases.

And this is shown below:

```
Accuracy of 0.8 =
93.0 %
Accuracy of 0.85 =
93.5 %
Accuracy of 0.9 =
94.0 %
Accuracy of 0.95 =
93.5 %
```

Step three: LDA algorithm:

ALGORITHM 20.1. Linear Discriminant Analysis

LINEARDISCRIMINANT ($\mathbf{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$):

1 $\mathbf{D}_i \leftarrow \{\mathbf{x}_j \mid y_j = c_i, j = 1, \ldots, n\}, i = 1, 2$ // class-specific subsets
2 $\boldsymbol{\mu}_i \leftarrow \text{mean}(\mathbf{D}_i), i = 1, 2$ // class means
3 $\mathbf{B} \leftarrow (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T$ // between-class scatter matrix
4 $\mathbf{Z}_i \leftarrow \mathbf{D}_i - \mathbf{1}_{n_i} \boldsymbol{\mu}_i^T, i = 1, 2$ // center class matrices
5 $\mathbf{S}_i \leftarrow \mathbf{Z}_i^T \mathbf{Z}_i, i = 1, 2$ // class scatter matrices
6 $\mathbf{S} \leftarrow \mathbf{S}_1 + \mathbf{S}_2$ // within-class scatter matrix
7 $\lambda_1, \mathbf{w} \leftarrow \text{eigen}(\mathbf{S}^{-1}\mathbf{B})$ // compute dominant eigenvector

The above figure shows LDA algorithms steps.

Further LDA explanation:

- We started by dividing the data into forty classes.
- Calculating the mean for each class and the overall mean of all classes.
- between-class scatter matrix was calculated by the following formula:

$$S_b = \sum_{k=1}^{m} n_k (\mu_k - \mu)(\mu_k - \mu)^T$$

note: multiplication between the 2 brackets is using outer product not dot product

- We centered each class by subtracting the mean of the class from it.
- Then using the centered data we compute the class scatter matrices.
- Calculate the sum of the class scatter matrices and its inverse.
- Multiply the inverse by the between-class scatter matrix
- Use the output to get the Eigen values and Eigen vectors.
- Using the first thirty nine dominant Eigen vectors we obtained the projection matrix.
- Then KNN was used to predict the labels of the testing data then we calculated the accuracy of the predicted labels.

<u>Tie breaking strategies:</u>

1- Choose the neighbor depending on the ordering of the training data.
2- Randomly select between tied neighbors.

Sklearn The classifier we use breaks the tie depending on the order.

*conclusion of the 2nd and 3rd step:

Accuracy of PCA=

```
Accuracy of 0.8 =
93.0 %
Accuracy of 0.85 =
93.5 %
Accuracy of 0.9 =
94.0 %
Accuracy of 0.95 =
93.5 %
```

Accuracy of LDA= 86.5 %

Step 3: Classifier tuning:

Applying the KNN algorithm for different values of k

Performance measure accuracy against the k for each alpha in PCA:

Classifying accuracy with respect to k for PCA (Alpha=0.8)

K	Accuracy
1	0.93
3	0.85
5	0.82
7	0.78

Classifying accuracy with respect to k for PCA (Alpha=0.85)

K	Accuracy
1	0.935
3	0.855
5	0.835
7	0.77

Classifying accuracy with respect to k for PCA (Alpha=0.9)

K	Accuracy
1	0.94
3	0.845
5	0.815
7	0.75

Classifying accuracy with respect to k for PCA (Alpha=0.95)

K	Accuracy
1	0.935
3	0.85
5	0.81
7	0.73

Performance measure accuracy against the k in LDA:

K	Accuracy
1	0.865
3	0.78
5	0.75
7	0.69

Step 4: Compare vs. Non-Face Images:

In this step we downloaded natural images dataset:

https://www.kaggle.com/prasunroy/natural-images

Reshaped it and converted it to grayscale to suit our model training.

We trained the model on classifying between 2 classes: Faces and Non-faces.

At first we used 200 images of each class and divided them in 2 sets:

Training set= 50 faces + 50 Non-Faces

Testing set=50 faces+ 50 Non-Faces

We then used the 2 algorithms LDA and PCA to train and test the model.

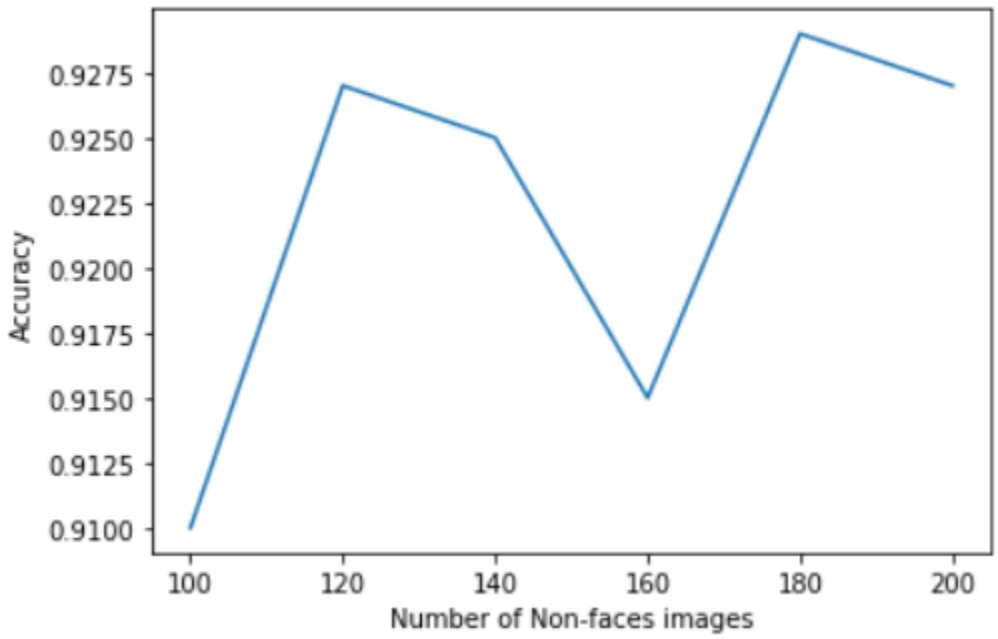
The PCA accuracy =

```
Accuracy of 0.8 =
91.0 %
Accuracy of 0.85 =
90.0 %
Accuracy of 0.9 =
90.0 %
Accuracy of 0.95 =
88.0 %
```

The LDA accuracy= 76.0 %

After that we started to increase the number of non-faces images while fixing the number of faces images.

<u>Relation between PCA accuracy and number or non-faces:</u>



Accuracy can be misleading if used with imbalanced datasets, and therefore there are other metrics based on confusion matrix which can be useful for evaluating performance.

The confusion matrix finds the distribution of all the predicted responses and shows how they compare to their true classes

Step 5(Bonus): splitting the dataset 70-30:

At this step we divided the dataset to 2 sets: training set that contain 70% of the data (7 images per person) and testing set that contain 30% of the data(3 images per person).

As a result of this splitting and applying the PCA and LDA algorithms the: the accuracy of each of them increased due to the increase of the number of training samples.

PCA accuracy=

```
Accuracy of 0.8 =
95.83333333333334 %
Accuracy of 0.85 =
95.83333333333334 %
Accuracy of 0.9 =
95.0 %
Accuracy of 0.95 =
94.16666666666667 %
```

LDA accuracy= 87.5%