Namrata Vivek Raghavan
nrgahav@purdue.edu
0027307068

Face Detection using PCA and LDA

In the following assignment, we are required to complete two different tasks. First, we are required to conduct face recognition using PCA and LDA for dimensionality reduction and then use nearest neighborhood rule for the classification.

Task 1

Face Recognition

Principal Component Analysis (PCA):

The principal component analysis is used to project a high dimensional data to a low dimensional representation. The following are the steps to conduct PCA:

1. Convert the images in training set from the RGB values to gray scale values.

2.Vectorize each of the 128 x 128 sized image into a 16384 x 1 vector $\vec{x_i}$, where i = 0, 1, 2 ... N-1 for N images in training set. Normalize the vectorized images

2. Calculate the global training mean

$$\vec{m} = \frac{1}{N}\Sigma_{i=0}^{N-1}\ \vec{x_i}$$

3. Create a matrix $X = [\vec{x_0} - \vec{m_0} \cdots \overrightarrow{x_{N-1}} - \overrightarrow{m_{N-1}}]$

4. Calculate the eigen vectors of the covariance matrix $C = X^T X$ and carry out its eigen value decomposition.

5. Let $\vec{u}$ be the eigen vectors of the covariance matrix in descending order of eigen values. The eigen vectors $\vec{w}$ are given by, $\vec{w} = X\vec{u}$. Normalize the $\vec{w}$ vectors.

6. Calculate a parameter K, where K is number of eigen values of eigen vectors $\vec{u}$ that are greater than 0.5.

7. Select the first K vectors in $\vec{w}$ to create the projection matrix $W_k = [\vec{w_0} \cdots \overrightarrow{w_{K-1}}]$.

8. Use the projection matrix to project the training images using the formula

$$\vec{y_i} = {W_k}^T(\vec{x_i} - \vec{m})$$

9. Create an array Z that provides a label to training images. Use the projected train data and its corresponding labels to train the nearest neighbor model. I have used the KNN model implementation through the scikit-learn library in python.

10.Vectorize each of the 128 x 128 sized test images image into a 16384 x 1 2. Repeat steps 2 and 3 for the test images.

Namrata Vivek Raghavan
nrgahav@purdue.edu
0027307068

11. Project the testing data using the method described and step 8. Obtain classification of the testing data using the KNN model.

12. Determine the accuracy of the of the predicted output from the test data by comparing it to the labels created.

13. Repeat the process for varying values of K.

Linear Discriminant Analysis (LDA)

The LDA process finds the eigen vectors $\vec{w}$ by maximizing the Fisher Discriminant Function given by,

$$J(\vec{w}\,) = \frac{\vec{w}^T S_B \vec{w}}{\vec{w}^T S_w \vec{w}},$$

where S_B is the between class scatter and S_w is the within class scatter

The between class scatter, S_B is given by the following formula,

$$S_B = \frac{1}{|C|}\Sigma_{i=1}^{|C|}(\vec{m}_i - \vec{m}\,)(\vec{m}_i - \vec{m})^T \,,$$

where $\vec{m}_i$ is the class mean and $\vec{m}$ is the global mean and C is the number of image in the dataset.

The within class scatter Sw is given by,

$$S_w = \frac{1}{|C|}\Sigma_{i=1}^{|C|}\frac{1}{|C_i|}\Sigma_{k=1}^{|C_i|}(\vec{x}_k{}^i - \vec{m}\,)(\vec{x}_k{}^i - \vec{m})^T,$$

where Ci is the number of images in a given class and $\vec{x}_k{}^i$ is the kth image vector in the ith class.

The Yu and Yang's algorithm is used to find the eigen vectors and the projection matrix, since S_w is singular in most cases.

Yu and Yang's algorithm

Due to the singularity of the S_w matrix the Yu and Yangs algorithm is used to calculate the eigen vectors and the projection matrix. Following are the steps in the Yu and Yangs algorithm to classify the images in the dataset:

1. Convert the images in training set from the RGB values to gray scale values.

2.Vectorize each of the 128 x 128 sized image into a 16384 x 1 vector $\vec{x_i}$, where i = 0, 1, 2 ... N-1 for N images in training set. Normalize the vectorized images

2. Calculate the global training mean

$$\vec{m} = \frac{1}{N}\Sigma_{i=0}^{N-1}\ \vec{x_i}$$

2. Calculate the training class mean, $\vec{m}_i = \frac{1}{Ci}\Sigma_{i=0}^{Ci-1}\ \ \vec{x_i}$, where Ci is the number of images in a given class.

3.Calculate the between class scatter SB, given by $S_B = \frac{1}{|C|}\Sigma_{i=1}^{|C|}(\vec{m}_i - \vec{m}\,)(\vec{m}_i - \vec{m})^T$, where $\vec{m}_i$ is the class mean and $\vec{m}$ is the global mean and C is the number of image in the dataset.

4. Carry out the eigen value decomposition of SB. Let V be the matrix consisting of the eigen vectors of SB in descending order. Y is a matrix of the first K eigen values in V. DB is the diagonal matrix

Namrata Vivek Raghavan
nrgahav@purdue.edu
0027307068

consisting of the first K eigen values of SB in descending order. The eigen value decomposition of SB is done by first conducting the eigen value decomposition of A = $\frac{1}{|C|}\Sigma_{i=1}^{|C|}(\vec{m}_i - \vec{m})^T(\vec{m}_i - \vec{m})$. The eigenvalues of SB are equal to those of A. Eigen vectors of SB are equal to eigenvectors of A times $(\vec{m}_i - \vec{m})$.

5. Calculate the matrix $Z = YD_B^{-0.5}$.

6. Conduct the eigen value decomposition of $Z^T S_w Z, where\ Z^T S_w Z = (Z^T X_w)(Z^T X_w)^T$. X_w is given by the following matrix. $X_w = [\overrightarrow{x_{00}} - \overrightarrow{m_0} \cdots \overrightarrow{x_{c-1k-1}} - \overrightarrow{m_{c-1}}]$

7. Let $\vec{u}$ be the eigen vectors of the $Z^T S_w Z$ matrix in ascending order of eigen values. Let $\widehat{U}$ be the smallest K eigen vectors of $\vec{u}$.

8. The projection matrix Wk is given by, $W_K^T = \widehat{U}^T Z^T$. Normalize the eigen vectors in Wk.

9.Use the projection matrix to project the training images using the formula

$$\vec{y_i} = W_k{}^T(\vec{x_i} - \vec{m})$$

10. Create an array Z that provides a label to training images. Use the projected train data and its corresponding labels to train the nearest neighbor model. I have used the KNN model implementation through the scikit-learn library in python.

11.Vectorize each of the 128 x 128 sized test images image into a 16384 x 1 2. 11. Project the testing data using the method described and step 9. Obtain classification of the testing data using the KNN model.

12. Determine the accuracy of the of the predicted output from the test data by comparing it to the labels created.

13. Repeat the process for varying values of K.

Results of Task 1:

Namrata Vivek Raghavan
nrgahav@purdue.edu
0027307068

Accuracy of PCA and LDA is calculated using the following formula

$$Accuracy = \frac{No\ of\ test\ images\ classified\ correctly}{Total\ number\ of\ test\ images}$$

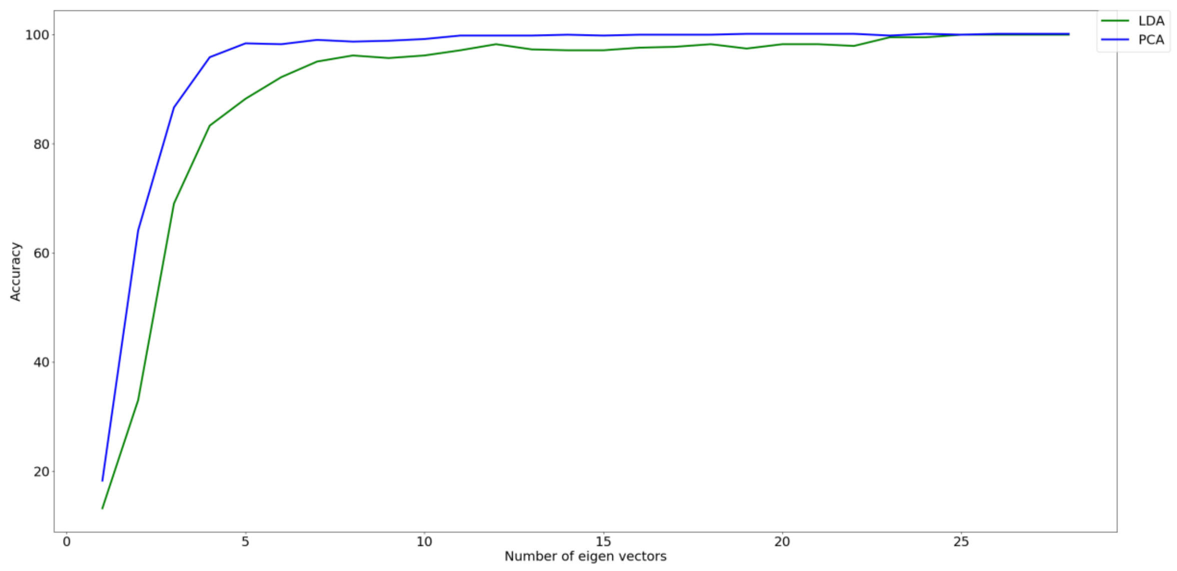

Figure 1: Plot showing the classification accuracy for both PCA and LDA

Comparison table between PCA and LDA

Subspace dimension	PCA accuracy	LDA accuracy
1	0.18253	0.13174
2	0.64126	0.33016
3	0.86666	0.69048
4	0.95873	0.83333
5	0.98413	0.88254
6	0.98254	0.92222
7	0.99048	0.95074
8	0.98730	0.96190
9	0.98889	0.95714
10	0.99206	0.96191
11	0.99841	0.97143
12	0.99841	0.98254
13	0.99841	0.97302
14	1.0	0.97143
15	0.99841	0.97144
16	1.0	0.97619
17	1.0	0.97777
18	1.0	0.98254
19	1.0	0.97460
20	1.0	0.98254
21	1.0	0.98254

Namrata Vivek Raghavan
nrgahav@purdue.edu
0027307068

22	1.0	0.97937
23	0.99841	0.99524
24	1.0	0.99524
25	1.0	1.0
26	1.0	1.0
27	1.0	1.0
28	1.0	1.0

Conclusion:

1. In the lower subspace dimensions the PCA is seen to have more accuracy than LDA.
2. While theoretically LDA is supposed to reach a 100 percent accuracy faster than PCA. In this experiment PCA is seen to reach a 100 percent accuracy at a smaller subspace dimension of 14 than LDA which reaches 100 percent accuracy at a subspace dimension of 24.