School of Computing
National University of Singapore
Biometrics Course
July 2017

Assignment #5: Face Recognition
Deadline: 31 July 5:00pm

Goal

The goal in this exercise is to use Principal Components Analysis (PCA) and Linear Discriminating Analysis (LDA) to recognize face images. Read all instructions first before coding. Some planning is required.

I. Data

Download face.zip in your computer and extract the images from face.zip. In this, you will find some face images taken from the CMU PIE database. There are 10 people, with each person captured under 24 different lighting conditions, for a total of 240 images. For convenience, these images have been split into two equal sets in the **train** and **test** directories.

The image files have names ppppp_xx_yy.bmp, where ppppp denotes the identity of the person; xx denotes the head orientation (all are frontal (xx=27) in this exercise); and yy denotes the lighting condition. All images have been cropped and aligned, and their height, width are 160 and 140 pixels, respectively.

II. Feature Extraction

Feature extraction plays an important role in face recognition. A good feature should be able to distinguish different users. In this part, you will learn to extract PCA feature and LDA feature for faces.

PCA feature

This part will guide you to learn a PCA feature extractor from a training set of image. You will also learn how to extract PCA feature with the extractor for an image.
Study the code in FisherFace.py. Use these functions for PCA feature extraction: myPCA, read_faces. Their purpose is described in the code. Follow the instructions:

1. **Read data**.Use read_faces to read in all the training images by setting the argument to be the path to **train** folder. You should now have a 22400 by 120 matrix called faces, whose columns are all the face images from the train directory, and an array of corresponding labels ranging from 0 to 9.

2. **Train PCA**. Using the myPCA function, compute the PCA projection matrix $\mathbf{W}$, the global mean vector $\mathbf{m}_e$, and the vector of eigenvalues. Read the code in myPCA

to understand how this is done. In particular, note the use of the inner product trick (as explained in the lecture notes) to avoid an "Out of memory" error.

3. **Select feature dimension**. Retain only the top K (K = 30) eigenfaces, by typing $\mathbf{W}_e = \mathbf{W}[:, :K]$.

4. **Project images**. For a face image $\mathbf{x}$, project it into PCA space using: $\mathbf{y} = \mathbf{W}_e^T(\mathbf{x} - \mathbf{m}_e)$. $\mathbf{y}$ is the pca feature representation of $\mathbf{x}$.

5. **Back-project PCA features**. As we know, PCA is also often used as a dimension reduction method. So we can recontruct an image $\mathbf{x}$ from a feature vector $\mathbf{y}$ in the learned PCA space by: $x = \mathbf{W_e y} + \mathbf{m_e}$.

LDA feature

This part will guide you to learn a LDA feature extractor from a training set of image. You will also learn how to extract LDA feature with the extractor for an image.
For the sake of computational cost, PCA is often used to reduce the dimension of inputs beforehand in LDA feature extraction. Use myLDA function in FisherFace.py for LDA feature extraction. In this task, we are actually implementing Whitened FLD in note lecture. Based on the results of PCA projection, follow the instructions bellow:

1. **Dimension reduction**. Reduce the dimension of input (22400) to K_1(K_1 =90). Retain only the top K_1 eigenfaces by $\mathbf{W}_1 = \mathbf{W}[:, :K_1]$, where $\mathbf{W}$ is the result of step 2 in PCA feature extraction part. Reduce the dimension of $\mathbf{x}$ by $\mathbf{x}' = \mathbf{W}_1^T(\mathbf{x} - \mathbf{m}_e)$. Apply the dimension reduction to whole input "faces". You should now have a K_1 by 120 matrix $\mathbf{X}$.

2. **Train LDA**. Apply myLDA on $\mathbf{X}$ and its corresponding labels and compute the LDA projection matrix $\mathbf{W}_f$ and centers of each class in LDA space $\mathbf{C}$. Read myLDA and understand how this is done.

3. **Project image**. For a given face $\mathbf{x}$, project it into LDA space by: $\mathbf{y} = \mathbf{W}_f^T\mathbf{W}_1^T(\mathbf{x} - \mathbf{m}_e)$.

III. Identification System

As we have learned, a face identification system has two sessions: enrollement and identification.

Enrollment Session

The first session to buid a face identification system is to enroll all users. That is, we need to store all users' face information in the system database. For simplicity, we can use the mean feature of each user to represent their identities. Based on the feature extraction methods in **Sec. II**, follow the instruction bellow:

1. Project all training images into corresponding feature space.

2. For each person, compute the mean **z** of all his feature vectors. Store these vectors as columns in a Numpy matrix **Z**. For convenience, it is best to store the columns of **Z** so that ith column corresponds to person with label i. Check that your **Z** matrix should have the size of D_f by 10, where D_f is the feature dimension. In our LDA task, **Z** is exactly the **Centers** returned by myLDA function.

Identification Session

Your face identification system is now ready. To identify a new face image **x** (re-shaped into a vector), first extract its feature by projecting it into feature space. Then search for the template that is closest to y, using the Euclidean distance metric. The index of the nearest template reveals the identity of **x**. For example, if the nearest template is column 2, the predicted id is **classLabel**[1].

You can now evaluate the performance of your identification system. Using the images in the test directory, identify each of them with your identification system as explained above.

The Confusion Matrix is a useful way to evaluate the performance a light identification system. Each element c_{ij} of an M by M Confusion Matrix **C** is the number of times an image from person **i** is identified as person **j** by the system. Thus the perfect identification system should produce a diagonal Confusion Matrix. Any non-zero off-diagonal element in C represents an error. The overall accuracy of the identification system can be calculated as the trace divided by the sum of all elements.

Calculate the 10 by 10 confusion matrix as follows. First, initialize all entries in the Confusion Matrix to 0. Then, for each test image, let predicted id be the identity that your classifier outputs, and let actual id be the actual identity of the test image (which you can determine from its filename). Add 1 to the entry in the actual_id-th row and predicted_id-th column of the Confusion Matrix.

IV. Task Sumary

Implement the following tasks in your codes:

1. Train a PCA feature extractor and build PCA-based Identifier using the training dataset. Evaluate the identifier with test dataset by Confusion matrix. (6)

2. Based on PCA task, visualize the mean face $\mathbf{m}_e$ and top 8 eigenfaces (principal components corresponding to the 8 largest eigen-values) in $\mathbf{W}_e$ as follows: re-shape each of them back into a 160 by 140 2D matrix using the Numpy function reshape, then use the Pyplot subplot command to create a 3 by 3 array of plots, and display the eigenfaces in these plots. Use the provided float2uint8 function to scale the pixel values to between 0 and 255. In the bottom right plot, display the mean face **m**. Entitle each plot with "Eigenface 1", "Eigenface 2", , "mean", so that it is clear which image is which. Submit these plots. (2)

3. Train a LDA feature extractor and LDA-based Identifier using the training dataset. Evaluate the identifier with test dataset by Confusion matrix. (6)

4. Based on LDA task, visualize the mean faces of each class using **centers** (denoted as $\mathbf{C}_f$) returned by myLDA function: Firstly, project center vectors from LDA space to PCA space: $\mathbf{C}_p = \mathbf{W}_f\mathbf{C}_f$; Secondly, project feature vectors fromp PCA space to image space: $\mathbf{C}_r = \mathbf{W}_e\mathbf{C}_p + \mathbf{M}$, where $\mathbf{M}$ is tiled from $\mathbf{m}_e$ to meet the dimension consistency; Finally, reshape each recontructed center image and plot them in 2 by 5 subplots with titiles as "Center1" ... "Center10". (3)

5. To improve the performance of identification system, fusion scheme is often adopted. Fusion scheme can be applied in three levels: raw data level, feature level and score level. In this task, plese implement fusion scheme in feature level. Based on PCA feature vector $\mathbf{y}_e$ and LDA feature vector $\mathbf{y}_f$, a fused feature vector can be constructed as $\mathbf{y} = \begin{bmatrix} \alpha\mathbf{y}_e \\ (1-\alpha)\mathbf{y}_f \end{bmatrix}$. Set α to be 0.5 and train a fusion-based Identifier using the training dataset and evaluate the identifier with test dataset by Confusion matrix. (4)

Based on the above tasks, answer the folloing questions:

1. Print the confusion matrix and overall accuracy for three identifiers. (3)

2. Compare the results for PCA feature and LDA feature, which feature is better? Why? (2)

3. Let $\alpha = 0.1, 0.2, \ldots, 0.9$. Retrain your identifier for fused feature and re-calculate its accuracy for each α. Plot accuracy versus α for different α. Submit this plot. What do you observe? (2)

4. Does the fused feature outperform both PCA feature and LDA feature? Why? (2)

Submission

Submit your code and answers to IVLE in a zip file.