School of Computing
National University of Singapore
Biometrics Course
July 2016

Assignment #5: Face Recognition

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.

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.

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

The main tasks for PCA are:

- Compute the PCA projection matrix $\mathbf{W}_e$ using the training images.
- Project all training images into PCA space, and represents each person (class) by the mean projected vector of his training images.
- Project all testing images into PCA space.

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 training images**. For each vector $\mathbf{x}$ in faces, 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. **Generate templates**. For each person, compute the mean $\mathbf{z}$ of all his pca representation vectors. Store these vectors as columns in a Numpy matrix $\mathbf{Z}$. For convenience, it is best to store the columns of $\mathbf{Z}$ so that ith column corresponds to person with label i. Check that your $\mathbf{Z}$ matrix should have the size of K by 10.

LDA feature

The main tasks for LDA are:

- Compute the LDA projection matrix $\mathbf{W}_f$ using the training images.
- Represents each person (class) by the centers of his training images in LDA space.
- Project all testing image into LDA space.

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 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. **Generate templates**. $\mathbf{C}$ are the templates for all people, with corresponding labels in ClassLabel from myLDA.

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

Feature fusion

In some cases, a face recognition method will fuse two features to improve the performance. The fusion is always performed by concatenating two features. Given an image, we are now able to get a PCA feature $\mathbf{y}_e$ and LDA feature $\mathbf{y}_f$. A new feature vector can be constructed by: $\mathbf{y} = \begin{bmatrix} \alpha\mathbf{y}_e \\ (1-\alpha)\mathbf{y}_f \end{bmatrix}$. Set $\alpha = 0.5$, compute the template of new feature for each person, and represent each test face by the new feature representation.

Classification

Your classifier is now ready. To classify a new face image $\mathbf{x}$ (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 $\mathbf{x}$. For example, if the nearest template is column 2, the predicted id is **classLabel**[1].

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

The Confusion Matrix is a useful way to evaluate the performance a classifier. Each element c_{ij} of an M by M Confusion Matrix $\mathbf{C}$ is the number of times an image from person $\mathbf{i}$ is classified as person $\mathbf{j}$ by the classifier. Thus the perfect classifier should produce a diagonal Confusion Matrix. Any non-zero off-diagonal element in C represents an error. The overall accuracy of the classifier 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.

Questions

Use PCA feature, LDA feature, fused feature, and perform the classification respectively. Answer folloing question.

1. Print the convolution matrix and overall accuracy for tree classifiers.
2. Compare the results for PCA feature and LDA feature, which feature is better? Why?
3. Let $\alpha = 0.1, 0.2, \ldots, 0.9$. Retrain your classifier 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?
4. Does the fused feature outperform both PCA feature and LDA feature? Why?

Submission

Submit your code and answers to IVLE.

Deadline: 24 July 5:00pm