# Homework 2

*Due: 11/11/2014, 11:50pm*

1. (45%) Programming PCA (Eigenface) with Matlab

Please unzip the face database file "PIE-NoLight". After unzipping it, you will see there are 65 folders. Each of them contains 21 face images of the same person. This face database is actually part of CMU-PIE database. In the original PIE database, it records face images of the same person under three variations: Poses, Illuminations and Expressions. That's why we call it "PIE" database. PIE-NoLight database is a sub-set of PIE database. It only contains 21 different "illumination variation" of the same person, and it collects images from 65 subjects. Therefore, there are 65 folders and each folder contains images from one individual. In every folder, there are 21 face images of different illumination.

A series of steps are designed in order to help you complete the goal of programming PCA with Matlab. Please follow the guide in the following paragraphs in sequential order, then you will be able to finish question 1 successfully.

(a) Let's start from person 1. Please read all 21 image files into Matlab workspace. You can use the function "imread()" to do it.

(b) As we discussed in class, usually we will represent one face image as a point in high dimensional space. Therefore, please convert each face images that you already read into workspace into a column vector.

(c) As we discussed in class, in PCA formula derivation, usually we will collect all of our data and arrange them into a big matrix I. Please do so. Remember, this matrix I is formed by filling its columns with each individual training sample.

(d) Compute the mean $\mu$ of all of the samples.

(e) Compute the "centered" data matrix $\mathrm{X} = \begin{bmatrix} | & | & \dots & | \\ I_1 - \mu & I_2 - \mu & \dots & I_N - \mu \\ | & | & \dots & | \end{bmatrix}$

(f) Now, let's apply Gram matrix trick. Please compute the eigenvalue/eigenvector of the Gram matrix, instead of covariance matrix. The definition of Gram matrix is

$$G = X^T X$$

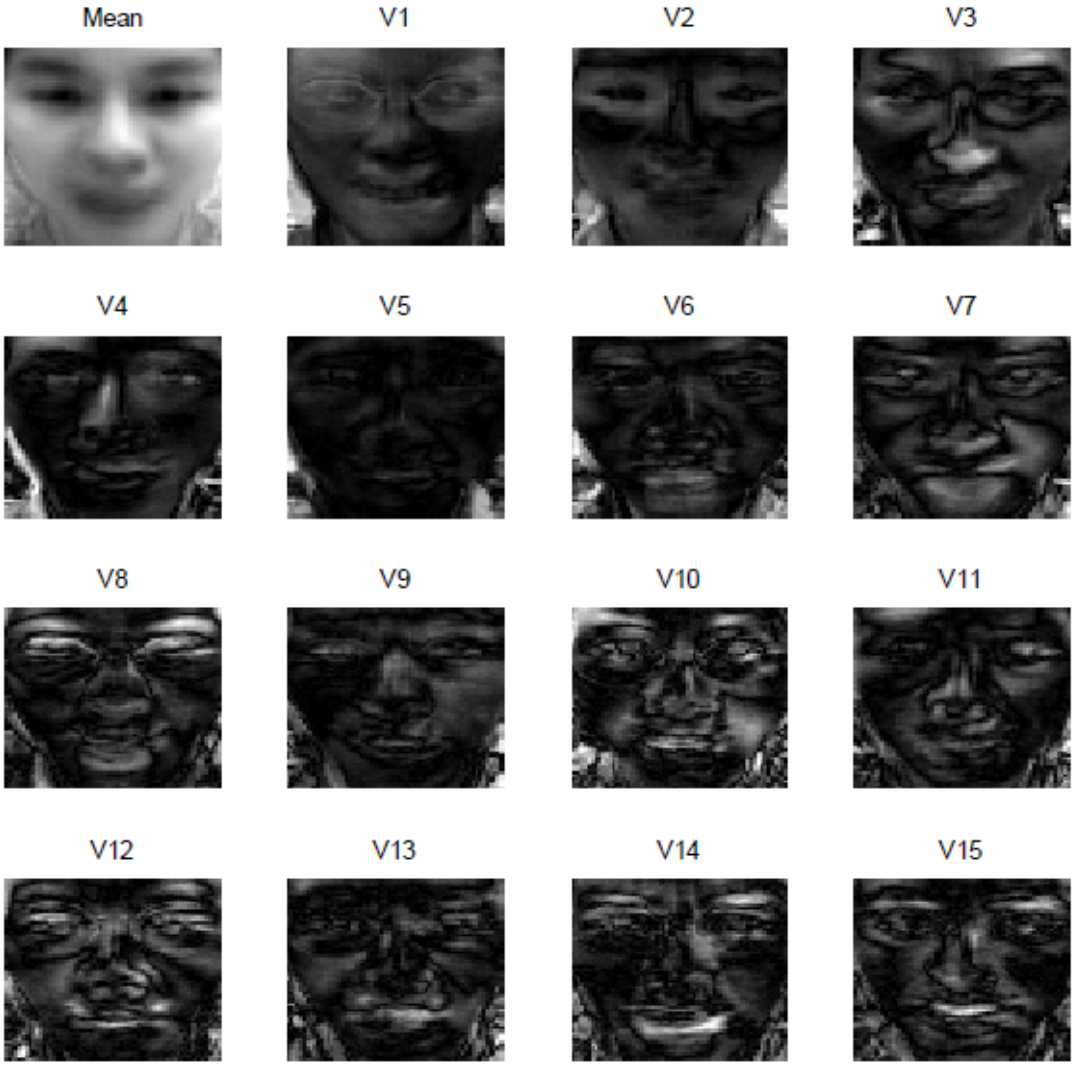
In Matlab, you should type: G=X'*X in order to compute G. Then use command [u D]=eig(G) to compute eigenvalue/eigenvector pairs of G.

After computing eigenvector set from Gram matrix, remember to convert it back to the real eigenvector set. You don't have to do anything for D, but for eigenvector set u, you need to use the command v=X*u to convert u to v which should be the "real" eigenvector for covariance matrix $\sum = XX^T$

(g) After you get the "real" eigenvector set v, you should normalize each eigenvector so that their length be 1.

(h) Then, you should sort your eigenvector according to the absolute value of eigenvalue, which is the diagonal element of matrix D

(i) In Eigenface problem, usually after computing eigenvector set, we would like to plot them out to see how they look like. Please write a Matlab codes to display the mean vector, and all other eigenface you computed, in one figure window, like this figure:




Please note that this is just an "example" figure in order to give an idea about the display format. Your final result (the eigenface you computed) should NOT look any similar to ANY of the faces of this figure. Also, how

many eigenface will you get? In this example figure there are 15. But in our problem, it is NOT.

Please write a Matlab script file, named as "displayEigenface.m". It should contain a function called "displayEigenface". And it should take the form as:

displayEigenface(pid)

where pid is the id of the subject, the range of pid is from 1 to 65. At run time, it should assume that there is a sub-directory "face" where all 65 directories are located. After executing it, it should read all images of person whose ID is pid, and compute its eigenface, and display the mean and all eigenface in a single figure window.

(j) Let's partition our data into two parts: training set and testing set.

- Training set: Let's use image index {7, 10, 19} from every class as our training data. Since there are 65 persons, the number of training images is 65*3=195.
- Testing set: Use all other images as testing set. The number of testing images is 65*(21-3)=1170.

Now, follow the description below …

(k) Train the set of eigenface from the training data. You should already know how to do it. After training, you should have three things:

A. A matrix V, in V, every column is an eigenface. Remember, you have to normalize every column of V to make them unit vector. Also, the order of the eigenface should reflect their importance. The 1st eigenface should correspond to the largest eigenvalue. The last eigenface should correspond to the smallest eigenvalue.

B. Mean vector, m.

C. A column vector D, where each element is the eigenvalue. The eigenvalue should also be sorted according to their absolute value.

Now, please do the following:

i. Project each of the training images to PCA subspace and record their coordinate (projection coefficient).

Suppose a training image is vectorized and stored in p. The projection coefficient of this image can be computed by pc=V'*(p-m)

ii. Project each of the testing images to PCA subspace and record their coordinate (projection coefficient).

iii. Now that you already have the projection coefficient of both training data and testing data. You can use those projection coefficients to perform face recognition.

For each testing image, suppose the projection coefficient is stored at

variable pc_test. You should compute the L2 Norm of the difference vector between pc_test and the projection coefficient of every training image. The recognition result is the label of the training image that gives the smallest L2 Norm. There are several way of doing it, one way is use the pseudo codes below:

```
%% store all projection coefficients in a 2 level cell array pc_train{}{}
%% Test…
minDist=inf;
For c=1:65
    For j=1:3
        D=norm(pc_test-pc_train{c}{j});
        If D<minDist
            minDist=D;
            classLabel=c;
        end
    end
end
```

Please perform face recognition for every testing image, and record their recognized label.

iv. Since you already know the real label (which is called “ground truth”) of every test image, you can compute the recognition rate. Please compare the recognized label with the ground truth label of every test image. Accumulate the number of “correct answer”. And recognition rate can be computed by #(correct answer)/#(testing images).

At the end of your script, please print out the recognition rate. You can use the function disp() and sprintf() to display formatted string, it is very similar to C/C++ output format. Please look them up in the help window of Matlab and find out how to use them by yourself. You should print out something like:

Recognition rate = 97.2%

Note that the number is not necessarily true. It is used as an example.

Please write a Matlab script file recog_pca.m to perform all steps in (j) and (k). Remember to print out the Recognition rate string at the end.

(l) Now, let’s perform parameter optimization for PCA. In previous practices, you already know how to use PCA to perform recognition. However, in PCA, there is a free parameter that we can optimize, which is the number of

eigenvector to use. If we use k eigenvectors for projection, each face image will be projected to k-dimensional space. Therefore, we can try to use different value of k to see that how many eigenvector we should use in order to have the highest recognition rate. Please follow the steps below:

i. Please package all the things you've done for question (j) and (k) into a Matlab function, with such format:

Function recogRate= recog_pca(k)

Here you can see that the function name is recog_pca, this function has one input argument k, which denotes how many eigenvectors to use for computing projection coefficient. The output is recogRate, which is a number between [0, 1] to indicate the recognition rate under such setting.

ii. Please use the function recog_pca(k) to fine-tune the parameter k, for the purpose of maximizing recognition rate.

Please write a Matlab script file OptimizeParam_pca.m to do it.

Overall, for question 1, the Matlab files you need to submit are:

- (15%) displayEigenface.m
- (15%) recog_pca.m
- (15%) OptimizeParam_pca.m

2. (55%) Implementing Fisher Linear Discriminant Analysis (FLDA) algorithm with Matlab.

Remember that in class we have discussed the FLDA is one of the most popular dimensionality reduction algorithms used in machine learning and pattern recognition. In this part, you are asked to implement FLDA algorithm with Matlab.

(a) Remember that in FLDA, we use training data from N classes (N>=2) to build our model. One important difference between PCA and FLDA, is that in PCA, we do not need "label" information for our data. We simply gather all training data, which may come from different class, and compute principal components without the need to know the "label" of each sample. But in FLDA, we need the "label" information. If you are not very sure about this part, please refer to the "Fisher LDA.pdf" which explained the detailed FLDA algorithm.

For this homework, again, we will use PIE_Nolight database. We will use the same "Training set" and "test set" as we used in Q1.

Let's define an input format for FLDA. This format has to incorporate the "label" information together with training data. There are many possibilities to do it. Let's use the "cell array" data structure in Matlab.

Please use the help window to look up how to create and use "cell array". Then, please organize the "training set" data with cell array. To be more specific, please write a Matlab code prepareCellData.m which accomplishes the following:

(1) Please create a cell array $X$. Suppose we have N classes, then $X$ should be a 1xN cell array.

(2) For class $i$ in training set, create a data matrix $D_i$ using the same format as you used in Q1. In other words, you should make each face image into a column vector. Then arrange them into a data matrix $D_i$

(3) Set $X\{i\}= D_i$

In this way, we have incorporated the label information in the structure of the cell array.

(b) Now, please write a Matlab function FLDA() to implement FLDA algorithm. This function should have following interface:

[W V M]=FLDA(X)

Where X is the data matrix we prepared in part (a); M is the global mean of the training data; V is the dimensionality reduction matrix and W is the LDA vectors where each column is a LDA component.

(c) Now, please use the training set data to train FLDA subspace. Then use that to perform classification on the "test set" data. To be more specific, please write a Matlab code classificationFLDA.m which achieves the followings:

(1) Organize the training data into the cell array X.

(2) Use part (b) to compute W, V and M.

(3) For each training sample, compute its coordinate in FLDA subspace. You should first subtract the global mean from the sample, then first project it to PCA subspace, then project it again to FLDA subspace. Please refer to "Fisher LDA.pdf" for more details.

(4) For each test sample, compute its coordinate in FLDA subspace too.

(5) Classify the test sample by using Nearest-Neighbor method with Euclidean distance in FLDA subspace.

(6) Please compute the overall classification rate and print it out like this:

"The overall recognition rate with FLDA is XXX%".

Overall, for question 2, the Matlab files you need to submit are:

- (15%) prepareCellData.m
- (25%) FLDA.m
- (15%) classificationFLDA.m

Please compress all Matlab *.m file into a single *.zip file. Please name this zip file with your student ID and your Chinese name. Submit it to TA through email before the deadline. You don't need to include the PIE-Nolight.zip file in your *.zip file.
You are very welcome to contact with TA if you have any question.
Good luck!

Prof. Yung-Hui Li