Assignment 5
Computer Vision 15–385, Spring 12
Due Date: Thursday 05/1/2012
Total Points: 90

In this assignment, you will be implementing a face recognition system using Eigenface.

1 Submitting Your Assignment

Your submission for this assignment will comprise of answers to a few theory questions, the code for your MATLAB implementation and a short writeup describing any thresholds and paramters used, interesting observations you made or things you did differently while implementing the assignment. The answers to the theory questions in Section 2 should be in a plaintext file or a pdf named `theory.txt` or `theory.pdf`. Each of the MATLAB functions you write as described in Section 3 along with any extra helper functions should be in a folder named `matlab`, upload only `.m` files to this folder and make sure all the files needed for your code to run (except data) are included. The writeup describing your experiments should be in a plaintext file or a pdf named `experiments.txt` or `experiments.pdf`. Also, you need to submit some `.mat` files and images files to show your intermediate and final results. Please refer to Section 3 and Section 4 . A directory has been created on for uploading all your course related files. The directory can be found at `/afs/cs.cmu.edu/academic/class/15385-s12-users/`*andrewid* (for graduate students, the folder is `15685-s12-users`. Inside your submission directory, create a subfolder named `p5` for this assignment. Do not add any extra layers of indirection to your directory structure. Your final upload should have the files arranged in this layout:

- `/afs/cs.cmu.edu/academic/class/15385-s12-users/`*andrewid*
  - `p5`
    - `theory.txt` or `theory.pdf`
    - `experiments.txt` or `experiments.pdf`
    - `result`
      - `trained.mat`
      - `eigface00.png`~`eigface10.png`
      - `face_x_y.png`
      - `accuracy.png`
    - `matlab`
      - `fn_double2img.m` (*already given*).
      - `pcaface.m`
      - `ploteigface.m`
      - `projectface.m`

- `reconstructface.m`
- `reconstruct_exps.m`
- `projecttrain.m`
- `recognizeface.m`
- `hw5Script.m`

You may need to run `aklog cs.cmu.edu` when you log in to be able to read and write from your submission directory. Your files are due by 23:59:59 on the submission day. Please make sure you have write permissions to your submission folder ahead of time so that problems (if any) can be fixed. We will be using timestamps to determine submission times and for late day counting, so do not modify your files after the submission deadline unless you wish to use a late day.

2 Theory Questions

Question 1: SVM (10 points)

Suppose that training examples are points in 2-D space. The positive examples are $X_+ = \{(1,1),(-1,-1)\}$. The negative examples are $X_- = \{(1,-1),(-1,1)\}$.

(a) Are the positive examples linearly separable from the negative examples? (i.e., Can you draw a line to separate the positive examples from negative examples)?

(b) Consider the feature transformation $\phi(x) = [1, x, y, xy]^T$, where x and y are the first and second coordinates of an example. Write down the transformed coordinates of X_+ and X_- (i.e., $\phi(X_+)$ and $\phi(X_-)$ for all four examples).

(c) Consider the prediction function $y(x) = w^T\phi(x)$. Give the coefficient w of a maximum-margin decision surface separating the positive from the negative examples. (hint: w is $[4\times 1]$ vector, whose elements are only 0 or 1).

Question 2: PCA (10 points)

Suppose that there are three points $\{(-2,-1),(0,0),(2,1)\}$ in 2-D space.

(a) Compute the first principal component.

(b) Project the original points into the 1-d subspace by the principal component you choose. Compute their coordinates in the 1-d subspace.

(c) Compute the variance of the projected data and the reconstruction error.

3 Programming

For this assignment, you will be using the ORL database of faces to explore the use of Principle Component Analysis (PCA) for representing and recognizing images. The image database is described at the following website.

http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

We provide following two data files in hw5.zip.

- orl_train.mat: It includes two variables: face_train and label_train. The face_train is the data for 360 face images. It is a (360×1) cell array, and the size of each cell is (88×72). You can display the i-th image using the following command.
  `> figure; imshow(uint8(face_train{i}));`
  label_train contains the labels of subject ID (1∼40) for all face images in face_train.

- orl_test.mat: It also includes two variables: face_test and label_test. The face_test is 40 test images organized in the same way as in face_train. The label_test are the labels for the face images in the face_test.

Part 1 Creating PCA basis vectors (15 points)

Write a program that creates the PCA basis vectors for a set of images.

`function [meanvec, basis] = pcaface(face_data)`

where face_data is the cell array containing the face images, meanvec is the (6336×1) mean of the images, and basis is a (6336× 100) PCA bases (or components). Each component is a (6336×1) column vector. Therefore, basis stores the top 100 components in descending order by their energy.

Run the function using face_train of orl_train.mat as the input of the function, and save its output meanvec and basis into **result/trained.mat**. Submit this file in addition to the program that you have written.

(*Hint*) The function pcaface performs the following tasks. (i) First transform face_train into a (6336×360) matrix. (ii) Compute the mean of face_train, which is a (6336×1) vector meanvec. (iii) Subtract the mean from each data point of face_train. (iv) Compute the (6336×6336) covariance matrix for the new face_train using the matlab function cov. (v) Obtain the top 100 eigenvectors of the obtained covariance matrix using the matlab function eigs. The (6336× 100) eigenvectors are the basis, which is the second output of this function.

You are not allowed to use any other matlab functions except cov and eigs.

Part 2 Plot the mean face (5 points)

Write a program that saves the mean face and the first K components (eigenfaces) as images.

`function ploteigface(meanvec, basis, K, imsize)`

The meanvec and basis are the output of **Part 1**, and imsize is (1×2) vector of the image size (e.g., [88 72]). Set K to 10. Then this function will save 11 image files **eigface00.png** to **eigface 10.png**. The **eigface00.png** is the mean face, which is the meanvec that is resized into [88 72] by using the provided helper function fn_double2img. **eigface01.png** ∼ **eigface10.png** are first K columns of basis that are also resized

by `fn_double2img`. Save these images into `result` folder and submit them along with the program, and explain what you see in the **`experiments`** file.

Part 3 Face projection (5 points)

Write a program that projects a face image into the PCA space and returns the resulting projection.

```
function fvec = projectface(faceimage, meanvec, basis, K)
```

The `faceimage` is a face image (88×72), K is the number of components to be used for projection ($1 \leq K \leq 100$), The output `fvec` is a ($K \times 1$) vector that represents the weights of the top K `basis` vectors.

(*Hint*) This function performs the following tasks. (i) Transform `faceimage` into a (6336×1) vector, from which the `meanvec` is subtracted. (ii) `fvec` is calculated by multiplying `basis(:,K)'` and the transformed `faceimage`.

Part 4 Face reconstuction (5 points)

Write a program that reconstructs the face image given a potentially cropped feature vector.

```
function reconimage = reconstructface(fvec, meanvec, basis, imsize)
```

The `reconimage` is a reconstructed face image (88×72). The program must accept a variable length of ($K \times 1$) `fvec`.

(*Hint*) This function performs the following tasks. (i) Multiply `basis(:,K)` and `fvec` where K is the length of `fvec`. (ii) It is added by `meanvec` and reshaped into the size of `imsize` by the matlab function `reshape`. The final result is the output `reconimage`.

Part 5 Testing Face reconstruction (10 points)

Pick the first 5 faces from the testing set `face_test`. Project each of them into the PCA space by using the function `projectface` of **Part 3** with 4 different K values: $K = \{5, 10, 50, 100\}$. Then reconstruct each of the 20 faces (i.e. 5 faces with 4 different K) by using the function `reconstructface` of **Part 4**. In your **`experiments`** file, compare the reconstructed faces to the original face qualitatively and quantitatively. Qualitatively explain what the differences are. Quantitatively measure the difference between the reconstructed faces and the original face by computing mean squared error (MSE) . Save each reconstructed face as **`face_x_y.png`**, where **x** is the face number (1-5), and **y** is the K used to reconstruct the face. Submit these images.

Write the function that automates this process as follows. `faceimages` is the (5×1) cell array for the face images, and `numcomps` = [5, 10, 50, 100].

```
function reconstruct_exps(faceimages, meanvec, basis, numcomps)
```

Part 6 PCA projection for all training images (5 points)

Write a program that computes the PCA projection for all 360 training images `face_train`.

```
function fvecs = projecttrain(face_train, meanvec, basis)
```

The `fvecs` vector should be (100×360), and `meanvec` and `basis` are output from **Part 1**. Note that this function calls `projectface` of **Part 3** iteratively for each image.

Part 7 Face recognition (20 points)

Write a program that performs face recognition.

```
function id = recognizeface(face_test, fvecs, label_train,
meanvec, basis, numcomp)
```

where `face_test` is all 40 test images from `orl_test.mat`, `fvecs` is the output from **Part 6**, and `label_train` is the training labels from `orl_test.mat`. The `meanvec` and `basis` are output from **Part 1**. The output `id` is the (40×1) recognized subject ID (1∼40) for each test image. `numcomp` is the number of components of the feature vector to use for comparison.

(*Hint*) This function performs the following tasks. (i) Run the function `projecttrain` of **Part 6** with `face_test`. Then, you will get (100×40) `fvecs_test`. (ii) Truncate the 100 dimensional feature vector to the `numcomp` length. (e.g. `fvecs(1:numcop,:)` and `fvecs_test(1:numcop,:)`). (iii) For each column of `fvecs_test`, find the closest column of `fvecs` by using Euclidean distance, and look up its `label_train`, which is going to be `id`. (e.g. If the closest column is 50 for the first test image, then `id(1) = label_train(50)`).

Compute the accuracy (% correct) as you vary `numcomp` = [1, 3, 5, 10, 25, 50, 100]. Plot a graph between the accuracy vs. `numcomp` and save it as **`accuracy.png`**. Also write the raw data in the **`experiments`** file.

4 Experiments

(10 points)

Submit a single script file **`hw5Script.m`** that consecutively runs **Part 1**, **Part 2**, **Part 5**, **Part 6**, and **Part 7**. We assign 10 points to the **`hw5Script.m`** and the submitted **`experiments`** file.