Computer Vision **CS 543 / ECE 549**

Homework 5
Due April 24, 2017

Answer the following questions and explain all solutions. Numbers in parentheses give maximum credit value.

1. Face Recognition (50%)

In this assignment you will implement the Eigenface and Fisherface methods for recognizing faces. You will be using face images from the Yale Face Database B where there are 10 faces under 64 lighting conditions. Using your implementation, you will evaluate the ability of the algorithm to handle lighting conditions of the probe images that differ from those in the training images.

You can download the data for this assignment at http://www.cs.ucsd.edu/classes/sp05/cse152/faces.zip

For more information on the Yale Face Database, see
http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html

Relevant papers:
http://www.cs.tau.ac.il/~shekler/Seminar2007a/PCA%20and%20Eigenfaces/eigenfaces_cvpr.pdf
http://www.cs.columbia.edu/~belhumeur/journal/fisherface-pami97.pdf

A. Recognition with Eigenfaces (PCA)

Step 1. Take each 50 × 50 pixel training image and vectorize into a 2500-dimensional vector. Then perform principal component analysis (PCA) on the entire set of training image vectors, retaining the first *d* principal components. Use the trick (discussed in class) for avoiding the full 2500x2500 covariance matrix. The *d* eigenvectors (when reshaped and displayed as images) are the Eigenfaces. You will display the top eigenfaces as images in your report.

Step 2. Now, for each of your training images, project to the d-dimensional Eigenspace. Once this is done, classification will be performed by nearest neighbor with the L2 (Euclidean distance) metric in the Eigenspace.

Step 3. Evaluate your algorithm on the frontal pose of the ten people in the Yale Face Database B. These images have been cropped and aligned for you. From the set of all ten individuals, we will consider 5 subsets, indexed as below:

Set 1. person*01.png to person*07.png
Set 2. person*08.png to person*19.png
Set 3. person*20.png to person*31.png
Set 4. person*32.png to person*45.png
Set 5. person*46.png to person*64.png

We have provided code "readFaceImages.m" to read in the images and store the person labels and subset numbers for each image.

What to report:

1) Train your Eigenface algorithm with $d = 9$ and $d = 30$ on all images in subset 1 (70 images). Then, evaluate your algorithm on subsets 1-5, and report the error rates in a table (error rate for each subset). For subset 1, you would expect perfect recognition because it is used for training. Include pseudocode (or equations) for the Eigenface recognition algorithm. (15%)
2) Display the top 9 eigenvectors (eigenfaces) after training. I suggest using `subplot(3,3,k)` with `imagesc` and `axis image, axis off, colormap gray` and save the figure as a png. (5%)
3) For both $d = 9$ and $d = 30$, display one original and the corresponding reconstructed face from each subset. The reconstructed face is constructed from the mean vector and from the eigenfaces and their coefficients. I suggest using `subplot(2, 5, k)` to display all five original and reconstructed faces in one figure. (5%)

B. Recognition with Fisherfaces (FLD)

Extend your Eigenface algorithm to perform the Fisherface algorithm discussed in class. First, project the training images via Eigenfaces to a $n - c$ dimensional space (n=number of images, c=number of different people); second, apply Fisher's Linear Discriminant to obtain $c - 1$ dimensional feature vector for each image.

What to report:

1) Evaluate the Fisherface algorithm. Train your Fisherface algorithm with $c = 10$ and $c = 31$ on all images in subset 1, and classify subsets 1-5 as in Part A. Report the error rates in a table (error rate for each subset). Use the same table as in Part A. Include pseudocode or equations for the Fisherface algorithm. (15%)
2) Now retrain PCA and FLD using both subset 1 and subset 5 as training data. Test on all subsets. (10%)

Your final table should look something like this:

Method (train set)	Subset 1	Subset 2	Subset 3	Subset 4	Subset 5
PCA (S1)					
FLD (S1)					
PCA (S1+S5)					
FLD (S1+S5)					

Notes:

1) You may not use built-in PCA functions like `pcacov` or `princcomp`. Use `eig` for computing the PCA and FLD subspaces. Be careful about whether the first or last columns correspond to the largest eigenvalues, as it depends on the way that you call `eig`.
2) You should pre-normalize each face by subtracting its mean and dividing by its standard deviation, as this will lead to better performance.
3) I encourage you to explore further on your own. Try different sized subspaces, or try training on different subsets and see how it impacts the others. Look at the smallest eigenvectors or at reconstruction error as the number of eigenvectors vary. You don't have to include these extra explorations in your report.

Acknowledgement: The data and basic design of this homework problem are from David Kriegman.

2. Object Categorization (50 pts)

In this problem you will be training a convolutional neural network (CNN) on the CIFAR-100 dataset. The problem description, submission instructions, and data are available on Kaggle at:

`https://inclass.kaggle.com/c/sp17-cs543-uiuc`

Please sign into Kaggle with your Illinois account in order to access the data. Due to time limits for processes on the EWS machines, we have provided everyone a virtual machine (VM) with MATLAB pre-installed on them. You can look up which VM is assigned to you here (note, it requires you to be logged into your Illinois email account to access): VM Assignments

Once you log into your VMs (through SSH, Putty, etc), you can launch MATLAB on the command line (e.g. matlab -nodisplay). The provided code also creates several figures used in the report. If you are using the VMs you may find it easiest to copy your final network to a local machine (e.g. an EWS machine) and create the figures there.

We have also installed Screen on the machines so you don't have to stay logged into them while your code is running. If you are unfamiliar with how to use Screen you can check out this tutorial. While we have provided code using MatCovNet to get you started, you are free to use any CNN framework you wish (e.g. Tensorflow, Caffe, Torch, etc).