CSE 152 – Introduction to Computer Vision – Homework 5
Instructor: Ben Ochoa
Due : Saturday, June 10, 2017, 11:59 PM

Instructions:

- Review the academic integrity and collaboration policies on the course website.
- This assignment must be completed individually.
- Submit your PDF report on gradescope. You must prepare a report describing your solution to the problem along with the produced results. The report must contain enough information for a reader to understand your methodology for solving the problem. Make sure you include the source code with comments under Appendix listing at the end of your report.
- Programming aspects of the assignments must be completed using MATLAB or Python.
- Submit your zipped code folder electronically by email to lmelvix@ucsd.edu, jsc078@ucsd.edu, and dperoni@ucsd.edu with the subject line **CSE152 Homework 5**. Make sure to include the **exact** subject line. The email should have one file attached. Name this file: CSE152_hw5_lastname_studentid.zip.
- All your source code should be in code folder with README.txt file explaining your code. For example, if you have **rotate_img.m**, your README.txt should have a detailed description: rotate_img.m: A function that takes an image matrix and degree as input and outputs the image rotated by the degree.
- The code is thus attached *both* as text in the appendix of the report and as m-files/py-files in the compressed archive.
- No physical hand-in for this assignment.

Introduction

In this assignment, we will have a look at some simple techniques for object recognition, in particular, we will try to recognize faces. The face data that we will use is derived from the Yale Face Database. For more information, follow link : YFDB-link. The database consists of 5760 images of 10 individuals, each under 9 poses and 64 different lighting conditions. The availability of such standardized databases is important for scientific research as they are useful for benchmarking different algorithms.

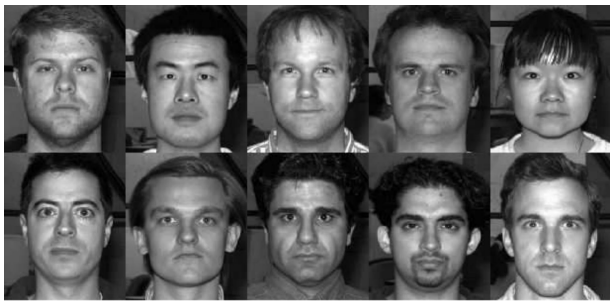

Figure 1: Figure 1: The Yale face database B.

In this assignment, we will only use 640 images corresponding to a frontal orientation of the face. These faces are included in the file **yaleBfaces.zip**. You will find the faces divided into five different

subsets. Subset 0 consists of images where the light source direction is almost frontal, so that almost all of the face is brightly illuminated. From subset 1 to 4, the light source is progressively moved toward the horizon, so that the effects of shadows increase and not all pixels are illuminated. The faces in subset 0 will be used as training images, and subsets 1 to 4 will be used as test images.

1 Naïve Recognition[15 points]

For this first problem, you will do face recognition by comparing the raw pixel values of the images. Let subset 0 be the train set, and report classification accuracy on test sets 1 to 4. Use the nearest neighbor classifier (1-NN) using the l_2 norm (i.e. euclidean distance).

1. (5 points) Once you have classified all images in a test set, report the average test error (percentage of misclassified test images). Report the average accuracy for each test set, i.e. **4** numbers.

2. (5 points) Comment on the performance, does it make sense? Is there any difference between the sets?

3. (5 points) show any two misclassified examples and explain why these might have been misclassified.

   Notes:

   - You can load a dataset by using e.g. `[trainset trainlabels]=loadSubset(0)` in `MATLAB` or `data,label=load_dataset([0], path)` in `Python`.
   - You can use `drawFaces.m` or`draw_faces.py` to look at the images e.g. `imshow(drawFaces(trainset, 10)))` in `MATLAB` or `draw_faces(data, 10)` in `Python`.

2 k-Nearest Neighbours Recognition [10 points]

Instead of using a single nearest neighbour, sometimes it is useful to consider the consensus (majority vote) of k-nearest neighbours. Repeat Part 1.1 using k-nearest neighbor classifier (k-NN).

1. (4 points) Test the performance for each test dataset using $k \in 1, 3, 5$. Note that you already have the results for $k = 1$ from Part 1.

2. (4 points) Test the performance for each test dataset using the l_1 norm rather than the l_2 norm with $k \in 1, 3, 5$.

3. . (2 points) Compare the performance of the classification with increasing k. Also comment on whether changing the distance metric influenced the results (if they did). Briefly justify the results observed.

3 Recognition Using Eigenfaces [25 points]

- (5 points) Write a function [W,mu]=eigenTrain(trainset,k) that takes as input a $N \times d$ matrix trainset of vectorized images from subset 0, where $N = 70$ is the number of training images and $d = 2500$ is the number of pixels in each training image. Perform PCA on the data and compute the top $k = 20$ eigenvectors. Return the $k \times d$ matrix of eigenvectors W, and a d dimensional vector mu encoding the mean of the training images. You should use the matlab command svd (or svds) to find the first k eigenvectors.

- (2 points) Rearrange each of the top 20 eigenvectors you obtained in the previous step into a 2D image of size 50×50. Display these images by appending them together into a 500×100 image (a 10×2 grid of images).
- (2 points) Explain the objective of performing PCA on the training images. What does this achieve?
- (2 points) Select one image per person from subset 0 (e.g., the 5 images `person01_01.png`, `person02_01.png`, ... , `person10_01.png`). Show what each of these images would look like when using only the top k eigenvectors to reconstruct them, for $k = 1, 2, 3, 4, 5, ...10$. This reconstruction procedure should project each image into a k dimensional space, project that k dimensional space back into a 2500 dimensional space, and finally resize that 2500 vector into a 50×50 image.
- (10 points) Write a function called `testlabels=eigenTest(trainset,trainlabels,testset,W,mu,k)` that takes as input :

  - The same N $\times$ d matrix trainset of vectorized images from subset 0
  - An N dimensional vector `trainlabels` that encodes the class label of each training image (e.g., 1 for person01, 2 for person02, etc.)
  - An $M \times d$ matrix `testset` of M vectorized images from one of the test subsets (1-4)
  - The output of PCA i.e. `W` and `mu`, and the number of eigenvectors to use k

  Project each image from `trainset` and `testset` onto the space spanned by the first k eigenvectors. For each test image, find the nearest neighbor (1-NN) in the training set using an L_2 distance in this lower dimensional space and predict the class label as the class of the nearest training image. Your function should return an M dimensional vector `testlabels` encoding the predicted class label for each test example. Evaluate `eigenTest` on each test subset 1-4 separately for values $k = 1...20$ (so it should be evaluated 4×20 times). Plot the error rate (fraction of incorrect predicted class labels) of each subset as a function of k in the same plot, and use the `MATLAB` or `Python legend` function add a legend to your plot.
- (2 points) Repeat the experiment from the previous step, but throw out the first 4 eigenvectors. That is, use k eigenvectors starting with the 5^{th} eigenvector. Produce a plot similar to the one in the previous step. How do you explain the difference in recognition performance from the previous part?
- (2 points) Explain any trends you observe in the variation of error rates as you move from subsets 1 to 4 and as you increase the number of eigenvectors. Use images from each subset to reinforce your claims.

4 Recognition Using Fisherfaces [20 points]

- (10 points) Write a function called `[W,mu]=fisherTrain(trainset,trainlabels,c)` that takes as input the same N $\times$ d matrix `trainset` of vectorized images from subset 0, the corresponding class labels `trainlabels`, and the number of classes $c = 10$. Your function should do the following :

  - Compute the mean `mu` of the training data, and use PCA to compute the first $N - c$ principal components. Let this be W_{PCA}.

- Use W_{PCA} to project the training data into a space of dimension $(N - c)$.
- Compute the between-class scatter matrix S_B and the within class scatter matrix S_W on the $(N - c)$ dimensional space from the previous space.
- Compute W_{FLD}, by solving for the generalized eigenvectors of the $(c-1)$ largest generalized eigenvalues for the problem $S_B w_i = \lambda_i S_W w_i$. You can use inbuilt functions to solve for the generalized eigenvalues of S_B and S_W.
- The fisher bases will be a $W = W_{FLD} W_{PCA}$, where W is $(c-1) \times d$ dimensional, W_{FLD} is $(c-1) \times (N-c)$ dimensional, and W_{PCA} is $(N-c) \times d$ dimensional.

- (5 points) As in the Eigenfaces exercise, rearrange the top 9 Fisher bases you obtained in the previous part into images of size 50×50 and stack them into one big 450×50 image.
- (5 points) As in the eigenfaces exercise, perform recognition on the `testset` with Fisherfaces. As before, use a nearest neighbor classifier (1-NN), and evaluate results separately for each test subset 1-4 for values $k = 1...9$. Plot the error rate of each subset as a function of k in the same plot, and use the `legend` function in `MATLAB` or `Python` to add a legend to your plot. Explain any trends you observe in the variation of error rates with different subsets and different values of k, and compare performance to the Eigenface method. paper-link