ECE 442 Introduction to Multimedia Signal Processing Winter 2021
Laboratory #3, KNN and Face Recognition, February 26, 2021
Report Submission Due Date: March 10, 2021

Note

1. Please submit in softcopy at eclass by 5pm of the due date, as a single zip file in FirstName_LastName_lab3.zip. The zip file is expected contain the following files: your answer sheet, Matlab code, and input/output files to be reproducible on our side.

2. Questions that should be answered in your lab report are numbered and marked as bold within the following text. Please number your answers accordingly. Cite any material that you have used (either it is a website or a paper or a piece of code).

3. Certain questions ask for images to be uploaded. Always use a lossless extension, e.g. png to save the images.

4. Make sure your Matlab code is bug-free and works out of the box. Please be sure to submit all main and helper functions. Be sure to not include absolute paths. Points will be deducted if your code does not work at our side.

1. Introduction:

In the previous session you learnt how to use PCA to map the data from a high dimensional space to a lower dimensional space. Due to the great number of features of each image (w*h), finding a feature extraction algorithm which is able to reduce the complexity of calculation is quite useful. On the other hand, the information of the images should be retained as much as possible to reduce the reconstruction error.

In this session you will learn how to detect face images and classify them to find whose face an unknown image might be.

1.1. First, you should load the training set into Matlab. The training set consists of 40 folders, and each folder contains 8 images of a distinct person. So, in total, you would have a training face dataset which consists of 320 images of 40 different persons. To load the dataset into Matlab, you should access the folders contained in “training” one by one and load all the images in each subfolder in Matlab.
The following lines of Matlab code displays all the images in training dataset:

```
folders = dir('training');
folders=folders(~ismember({folders.name},{'.','..'}));
subFolders = folders([folders.isdir]);

for k = 1 : length(subFolders)
   cur_dr=['training\' subFolders(k).name];
   images=dir(cur_dr);
   images=images(~ismember({images.name},{'.','..'}));
   for i=1 : length(images)
     imshow(imread([cur_dr '\' images(i).name]));
   end
end
```

Hint*: If you are using Linux change "\" to "/".

2. Determining the best lower dimensional sub-space:

In the previous session, you learnt to find **K** high-variance eigenvalues and their correspondent eigenvectors of the images. By ignoring the remnant low-variance eigenvectors, we are able to map images into a lower dimensional space.

2.1. Calculate the eigenvalues of the covariance matrix of the dataset matrix using the Matlab script you wrote last session.

***Hint: for all question of this lab use the second method (SVD) to get the eigenvalues.**

Question 1: Sort the eigenvalues in descending order and plot them (if you are using SVD function of Matlab they are already sorted). Look at the decay of the eigenvalues. How would you choose K? (5 point)

3. Distinguishing face image from others:

3.1. Find the contribution of eigenfaces (from 1 to K) for each train image. (Choose K=50). Compute the contribution of each of the eigenfaces for each train image based on:

$$W_{50*1} = (V_{10304*50})^T (x_{10304*1} - E\{X\}_{10304*1}) \qquad \text{Eq.1}$$

Where x is each train image and V is the matrix of eigenfaces (U matrix of SVD). Finally, you should have a vector of weights (50*1) for each train image. Calculate the mean of these vectors (W_{Mean}; $its\ size\ should\ be\ 50*1$)

3.2. Load 'arctichare.png'and resize the grayscaled image to have the same dimensions as the faces database (112*92).

3.3. Find contribution of eigenfaces for 'arctichare.png' using Eq.1 ($W_{artichare}$).

Question 2: Calculate the Euclidean distance between $W_{artichare}$, W_{mean} ($d_{artichare}$). Now pick an image from the faces database test folder (e.g. "9.png" from the first subject). Find eigenfaces weights for the test image using Eq.1 (W_{test}). Calculate the Euclidean distance between W_{test}, W_{mean} (d_{test}). Compare $d_{artichare}$,d_{test}. What do you observe? (15 points)

Question 3: Find the vector of weights for 3 other test samples and find the Euclidean distance between their weight vector and W_{mean}. Pick a reasonable threshold to be able to distinguish face images from other images (if $d > threshold \rightarrow$ the test image is not a face image). (10 points)

4. Face Recognition:

In this part we are going to classify unseen test face images. Assume that we are given a test image. The idea is to find the mean weight vector for training images of each person and find the closest to the weight vector of the test image.

4.1. You had calculated the vector of weights for each train image in part 3.1. Calculate the mean of weights for each person (for example for the first person you should take the average of weight vectors for the 8 training images of him). Now you should have 40 set of weight matrices. ($W_{p-mean1}$,$W_{p-mean2}$,$W_{p-mean3}$,…,$W_{p-mean40}$)

Question 4: Construct the faces with the eigenface contribution weight matrix of $W_{p-mean1}$ and $W_{p-mean40}$using the reconstruction formula (Eq.2) and plot them. Do they look like the first and the last subjects respectively? What attributes do the resulted images have in common with the original images of subjects (hair style, rotation of head, illumination, …). Attach the images to the report(15 points)?

$$\hat{x}_{10304*1} = V_{10304*50} W_{50*1} + E\{X\}_{10304*1} \quad (Eq.2)$$

Hint*: $E\{X\}_{10304*1}$ is the mean image (average of the all training images)

4.2. Load the test dataset which consists 2 different images for each subject.

4.3. Calculate eigenfaces weights matrix for each of these test images using Eq.1.

Question 5: Calculate Euclidean distance of the first test sample weight matrix with each of $W_{p-mean1}$,$W_{p-mean2}$,$W_{p-mean3}$,…,$W_{p-mean40}$. Report the W_{p-mean} with the shortest distance and declare its index as the predicted label of the first test sample. Do the same thing for other 79 test images. Report the accuracy of prediction over all 80 test images. (10 points)

Now we are going to use the KNN algorithm to find the labels of the test samples. The idea of KNN comes from the fact that similar examples are more likely to have similar labels. Thus, given a new test sample we would be able to find its label based on how similar to the train samples it is. KNN finds the K closest train samples to the test sample and returns the majority class label of the K

samples as the predicted label of the test sample. Here, we use Euclidean distance to find how close two samples are to each other.

4.4. You had calculated the weight vector for each of the train images in part 3.1 and weight vectors of the test samples in Question4. Assume that you want to predict the label of the first test image. You should calculate the Euclidean distance between its correspondent weight vector and all of 320 weight vectors of train images. Taking majority voting between the K closest train weight vectors labels gives you the label of the test image.

For example consider that you are testing an image and the weight vector of the 4th image of the 5th person, 7th image of the 5th person, and 2nd image of the 37th person are the 3 closest weight vectors (K_{KNN}=3). Then, the predicted label of the test image would be 5. As another example consider that you are going to test another image and the weight vector of the 1st image of the 24th person is the closest weight vector (K_{KNN}=1). Then, the predicted label of the test image would be 24.

Question 6: Calculate the Euclidean distance between the first test weight vector and all train vectors. Find the K shortest distances (Assume that K=1) and report the mod of the K samples labels as the predicted label of the first test images. Do the same thing for other 79 test images. Report the accuracy of prediction over these 80 test images. Compare the accuracy with accuracy of Question5. Which one of these methods do you prefer for face classification? Why? (25 points)

4.5. Take your own photo, crop it and resize it to have the same dimension as the images of the dataset. Convert it to a grayscale image.

Question 7: Calculate eigenfaces weights matrix for your photo and compare it with $W_{artichare}$, ("artichare" image weight vector) and W_{mean}(training set mean weight vector) that you had calculated before. What do you see? Treat your photo as a test image and test it using the implemented KNN algorithm. What similarities do you see between your photo and the predicted class image (Age, gender, glasses, hairstyle, …)? (20 points)

Bonus:

5. In the previous part we assumed that K=1. But is it really the best choice?

Question 8: Evaluate accuracy of the KNN algorithm when $K_{KNN} \in \{3,5,7\}$. What is the best choice for K_{KNN} (10 points)?