ECE 442 Lab Report 3:

KNN and Face Detection

Section: H51

Compiled by: Abhi Sharma

ID: 1643951

NOTE

All the code can be found in the one single Lab3.mlx file attached with the zip file.

1. Introduction

This section contains the code we implemented as part of the last lab.

Code:

```
%The following is the code from last lab
folders = dir('Face\training');
folders = folders(~ismember({folders.name}, {'.', '..'}));
subFolders = folders([folders.isdir]);
mat = []
mean_mat = []
for K = 1:length(subFolders)
    cur_dr1 = ['Face\training\' subFolders(K).name];
    images = dir(cur_dr1);
    images = images(~ismember({images.name},{'.','..'}));
    for i = 1:length(images)
        img_mat = imread([cur_dr1 '\' images(i).name]);
        img_mat = img_mat(:, :, 1);        %turning the image into a
        img_mat = reshape(img_mat, [], 1);
        mat = [mat img_mat];
    end
end
mean_mat = mean(mat, 2);
% Making our data X
im2 = reshape(mean_mat, [112, 92])
im3 = cast(im2, "uint8")
lol = size(im3)
imshow(im3)
imwrite(im3, "mean.bmp")
%Subtract the mean image from all the samples
data_X = []
```

```
for K = 1:320
    %temp = reshape(mat(:, K), [112, 92]);
    subs = cast(mat(:, K), "double") - mean_mat;
    d_subs = size(subs);

    data_X = [data_X subs];
end
%2.5
size(data_X)
trans_data_X = data_X'
T = (trans_data_X*data_X)/320
size(T)

[U, D, V] = svd(data_X)
eig_faces_SVD = U
eig_val_SVD = (diag(D).^2)/320
eig_6_max_vec_SVD = []
[eig_val_SVD_sorted, eig_val_indices] = sort(eig_val_SVD, 'descend');

```

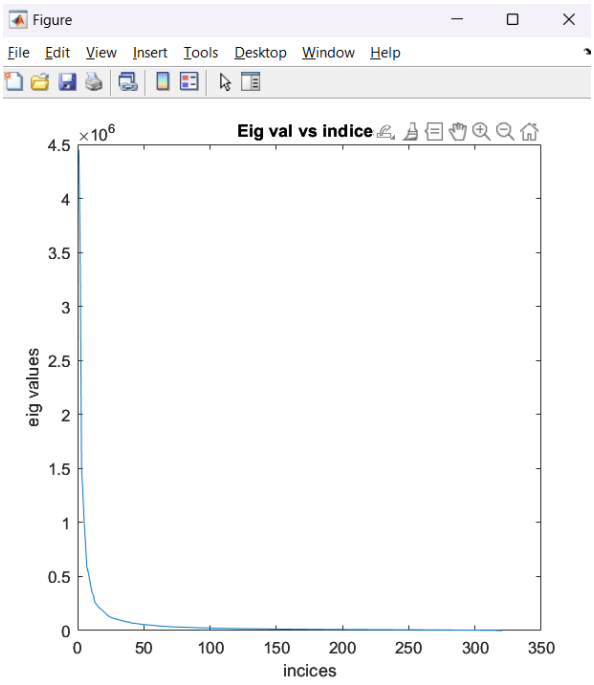
2. Determining the best lower dimensional subspace.

Code:

```
% Part 2
%Q1 We need to plot the eigenvalues vs the index
plot(eig_val_indices, eig_val_SVD_sorted)
ylabel("eig values")
xlabel("incices")
title("Eig val vs indices")
%set(gcf,'Position',[100 100 500 500])
% Now the question remains that how do we choose the right value for K
% We will choose the point on the curve where the eigenvalues start to
% level off. Looking at out curve, I identified that the appropriate value
% for K should be 22
set(gcf,'Position',[100 100 500 500])
```

Q1. Plot eigenvalues vs index. Look at the decay of the eigenvalues. How would you choose K? (2 point)

Ans.



The figure above shows the curve that we obtain after plotting the eigen values against the respective indices.

Looking at the curve, we can clearly notice that the starting few eigen vectors with the highest eigen values have the most contribution in the contrast and the details of the image. Now when it comes to choosing the appropriate value of k, we can

choose values between 30 and 150 (depending on how much complex we want our data to be as it would increase the time and space complexity of our model). I personally think that k = 40 should be a reasonable enough values as it gives the right balance between the right amount of detail in that data and the complexity of the model.

3. Distinguishing face images from others

Code:

```
% Part 3
%Reconstruction using 16 eigenfaces
% Looking at the curve I figured that the eigenvalue saturaates at around
% the 100 mark and thus the eigenvectors beyond that point don't contain a
% lot of value beyond that point (ofcourse we can also try to reduce this
% value if we are looking at the computational/time trade-off.
eig_seventy_face = U(:, 1:40);
eig_seventy_face_t = eig_seventy_face'
W_mat = []
for K = 1:length(subFolders)
    cur_dr1 = ['Face\training\' subFolders(K).name];
    images = dir(cur_dr1);
    images = images(~ismember({images.name},{'.','..'}));
    for i = 1:length(images)
        img_mat1 = imread([cur_dr1 '\' images(i).name]);
        img_mat1 = img_mat1(:, :, 1);        %turning the image into a greysid
        img_mat1 = reshape(img_mat1, [], 1);
        img_mat1 = cast(img_mat1, "double");
        W = eig_seventy_face_t*(img_mat1-mean_mat);
        W_mat = [W_mat W];
    end
end
```

```
size(W_mat)
W_mean = mean(W_mat, 2)
arctic = imread("arctichare.png");
arctic = arctic(:, :, 1)   %Turning the image into a greyscale image
arctic = imresize(arctic, [112, 92]);
arctic = reshape(arctic, [], 1);
arctic = cast(arctic, "double");
W_arctic = eig_seventy_face_t*(arctic-mean_mat);
% Eucledian distance between W_arctic and W_mean
eud_dis_arctic = norm(W_arctic- W_mean)
rand_img = imread("Face\testing\s5\9.png")
rand_img = rand_img(:, :, 1);
rand_img = reshape(rand_img, [], 1);
rand_img = cast(rand_img, "double");
W_test = eig_seventy_face_t*(rand_img - mean_mat);
eud_dist_test = norm(W_test - W_mean)
% Comparing the above eucledian distances, we can clearly see that the
% eucledian distnce dtest is lower than the eucledian distance darctichare
```

```
%Question 3
%The for loop below takes four images form the testing dataset and
%then finds the W_test and the respective eucledian distance for all of the
%4 images
k = 1;
euc_dist = [];
W_test_i = cell(1, 4);
j = 9;
for i = 2:5
    name_of_file = sprintf("Face\\testing\\s%d\\%d.png", i, j);
    r_i = imread(name_of_file);
    r_i = r_i(:, :, 1);
    r_i = reshape(r_i, [], 1);
    r_i = cast(r_i, "double");
    W_test_s = eig_seventy_face_t*(r_i - mean_mat);
    W_test_i{k} = W_test_s;
    euc_dics = norm(W_test_s - W_mean);
    euc_dist = [euc_dist euc_dics];
    k = k+1;
end
%The following line display all the Wtest for the four images
W_test_1 = W_test_i{1}
W_test_2 = W_test_i{2}
W_test_3 = W_test_i{3}
W_test_4 = W_test_i{4}
disp(euc_dist)         %This line displays all the eucledian distancesfor the
                       % four Weights as an array

```

Q2. Calculate the Euclidean distance between $\boldsymbol{Warctichare}$, $\boldsymbol{Wmean}$ ($\boldsymbol{darctichare}$). Now pick an image from the faces database test folder. Find eigenfaces weights again using Eq.1 and form the matrix of weights ($\boldsymbol{Wtest}$). Calculate the Euclidean distance between $\boldsymbol{Wtest}$, $\boldsymbol{Wmean}$ ($\boldsymbol{dtest}$). Compare $\boldsymbol{darctichare}$, $\boldsymbol{dtest}$. What do you observe? (3 points)

Ans. The following was the Euclidian distance between Wtest(weight vector of the random image) and Wmean (dtest):

```
eud_dist_test = 4.2596e+03
```

And the following is the Euclidian distance between darctichare and dtest (darctichare):

```
eud_dis_arctic = 9.7603e+03
```

We can clearly see that the Euclidian distance $\boldsymbol{darctichare}$ is way larger that the Euclidian distance dtest. This also makes sense as the arctichare is not a face image and thus it is very different from the average of the weight matrices for all the training samples.

Q3. Find the matrix of weights for 4 other test samples and find the Euclidean distance between the matrices and $\boldsymbol{Wmean}$. Pick a reasonable threshold to be able to distinguish face images from other images. (2 points)

Ans. The code can be found above. The following picture shows an array with the Euclidian distances for the four test samples.

```
1.0e+03 *

 3.4300    3.6500    3.1578    4.2596
```

For a reasonable threshold, I believe that anything with a Eucledian distance less than 4500 should be classified as a face image. However, this range might change if we have a lot of anomalies in the data and therefore to be safe we can take a range for the threshold between 4500 and 5000.

4. Face Recognition using KNN:

Code:

```
% % Question 4
% We need to calculate the mean weights for each and every person in the
% dataset
% We had stored the all the weights of each image in the matrix called
% W_mat, so now all we need to do is to access all that data in multiples
% of 9 and the find the mean and we can easily achieve this using a nested
% loop
W_p_mean_tot = [];
 for i = 1:length(subFolders)
     W_i_tot = [];
     W_p_mean = [];
     recost_W_p = [];
     reconstructed_W_p = [];
     for j = 8*(i-1)+1:8*(i-1)+8
         W_i = W_mat(:, j);
         W_i_tot = [W_i_tot W_i];
     end
     W_p_mean = mean(W_i_tot, 2);  % reconstruct this
     recost_W_p = (eig_seventy_face * W_p_mean) + mean_mat;
     reconstructed_W_p = reshape(recost_W_p, [112, 92]);
     reconstructed_W_p = (reconstructed_W_p - min(reconstructed_W_p, [], 'all')) / (max(reconstructed_W_p, [], 'all') - mi

     if i<=5
         imshow(reconstructed_W_p)
         imwrite(reconstructed_W_p, sprintf("test_%d.bmp", i))
     end
%      imwrite(reconstructed_mat_img, sprintf("Q8rec.bmp"))
%      plot_img = imresize(W_p_mean, [112, 92]);
%      plot_img = cast(plot_img, "double");
%      imshow(plot_img)
      W_p_mean_tot = [W_p_mean_tot W_p_mean];    %This matrix contains the mean Weight matrix for each person

 end
```

```
% Question 5
% Loading the test dataset
% I am using tic/tac to measure the time it takes to run on our testing
% dataset
folders_t = dir('Face\testing');
folders_t = folders_t(~ismember({folders_t.name}, {'.', '..'}));
subFolders_t = folders_t([folders_t.isdir]);
test_mat = [];
W_test_tot = [];
actual = [];
euc_dist_Q5_tot = [];
for K = 1:length(subFolders)
    cur_dr1 = ['Face\testing\' subFolders(K).name];
    images = dir(cur_dr1);
    images = images(~ismember({images.name},{'.','..'}));
    for i = 1:length(images)
        actual = [actual K];
        img_mat2 = imread([cur_dr1 '\' images(i).name]);
        img_mat2 = img_mat2(:, :, 1);        %turning the image into a greyside b
        img_mat2 = reshape(img_mat2, [], 1);
        img_mat2 = cast(img_mat2, "double");
        test_mat = [test_mat img_mat2];
        W_test = eig_seventy_face_t*(img_mat2 - mean_mat);
        W_test_tot = [W_test_tot W_test];
    end
end
```

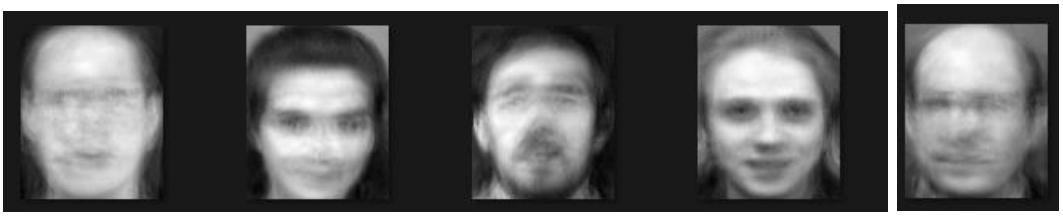
```
% Now since we have found the eucledian distances for all the images we need
% to index with the min eucledian distance foa lll 80 images and then we
% can calculate the accuracy of our model
tic
min_preds_index = [];
min_preds_euc = [];
for l = 1:80
    euc_dist_Q5_tot = [];
    for g = 1:40
        euc_dist_Q5 = norm(W_test_tot(:, l) - W_p_mean_tot(:, g));
        euc_dist_Q5_tot = [euc_dist_Q5_tot euc_dist_Q5];
    end
    [min_euc_dist, index_eud_dist] = mink(euc_dist_Q5_tot, 1);
    min_preds_index = [min_preds_index index_eud_dist];
    min_preds_euc = [min_preds_euc min_euc_dist];
end
% The above code gives us both the index and the minimum eucledian distance
% for all the values. Now we can calculate the accuracy of our model
acc_Q5 = sum(min_preds_index == actual)/80; % this is the accuracy for Q5
toc
str = sprintf("Accuracy of model in Q5 is %0.4f", acc_Q5);
disp(str)
```

```
% Question 6
% In this part of the lab we shall use KNN to find the label of each test
% image.
% In part three of this lab we had made a weight vector for all the test
% images names Wmat
% Now we need to calculate the eucledian distance of the weight vector of
% every test sample from the weight wevtor of every image and the choose
% and then do the majority voting
tic
K = 5; %Note K here represents the K shortest distances we are considering
knn_preds = [];
for a = 1:80
    euc_dist_Q6_tot = [];
    for b = 1:320
        euc_dist_Q6 = norm(W_mat(:, b) - W_test_tot(:, a));
        euc_dist_Q6_tot = [euc_dist_Q6_tot euc_dist_Q6];
    end
    [K_min_euc_dist_Q6, index_k_min_Q6] = mink(euc_dist_Q6_tot, K);
    most_freq = mode(ceil(index_k_min_Q6/8));
    % Calculating the index of the person
    %in the above statement I am calculating the mode because I want to
    %know which element has the most frequency. I alst divide the index by
    %8 and the round it to the next integer to get the correct index
    knn_preds = [knn_preds most_freq];
end
acc_knn = sum(knn_preds == actual)/80;
str_Q6 = sprintf("The accuracy of KNN model is %0.4f", acc_knn);
disp(str_Q6)
toc
```

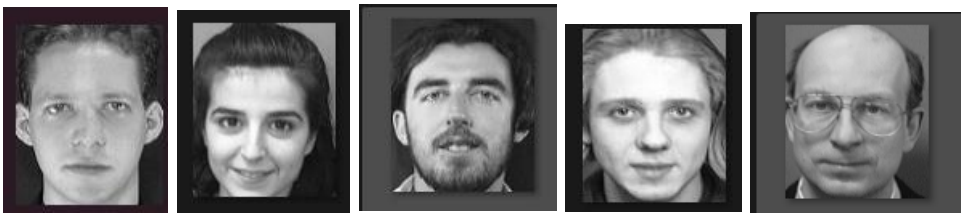
```
%Q7 In this section of the lab I'm trying to test to see what subject does my
%face match to the most
K = 5;
my_img = imread("lol1.jpg");  %This is my image that I'm reading from the directory
my_img_lin = reshape(my_img, [], 1);
my_img_lin = cast(my_img_lin, "double");
W_my_face = eig_seventy_face_t*(my_img_lin - mean_mat);
euc_dist_my_face_tot = [];
for p = 1:320
    euc_dist_my_face = norm(W_my_face - W_mat(:, p));
    euc_dist_my_face_tot = [euc_dist_my_face_tot euc_dist_my_face];
end
[K_min_euc_dist_Q7, index_k_min_Q7] = mink(euc_dist_my_face_tot, K);
pred_my_face = mode(ceil(index_k_min_Q7/8));
disp("My face was predicted as subject s31 or 25 according to the subfolder variable");
```

Q4: Construct the faces with the eigenface contribution weight matrix of ($Wp–mean1$,$Wp–mean2$, … ,$Wp–mean5$) and plot them. Do they look like the first 5 subjects face images? What attributes do the resulted images have in common with the original images of subjects (hair style, rotation of head, illumination, …) (3 points)?

Ans. The code for this can be found above. The following are the five images obtained. The 5 images can also be found attached with the zip file (I had use imwrite to write the images to the directory).



And the following are the actual (sample) images:



The images produced look very similar to the actual images. For every single corresponding images above, we can clearly notice that we can retain a lot of details. For instance, we can distinctly identify male subjects from the female

subjects just by looking at the hair styles. For the final sample we can clearly notice the baldness in the produced version of the images. Similarly we can also get an idea of other characteristics like rotation of head and illumination as well.

Q5: Calculate Euclidean distance of the first test sample weight matrix with each of $Wp\text{–}mean1$,$Wp\text{–}mean2$,$Wp\text{–}mean3$, … ,$Wp\text{–}mean40$. Report the $Wp\text{–}mean$ with the shortest distance and declare its index as the label of the first test sample. Do the same thing for the other test images. Report the accuracy of prediction over these 80 test images. (3 points)

Ans. The code for this can be found above and in the attached file as well. After performing everything I got an overall accuracy of 88.77%. I also displayed the accuracy in the .mlx file as well.

```
Elapsed time is 0.029341 seconds.

Accuracy of model in Q5 is 0.8875
```

Q6: **Find the K shortest distances (Assume that K=5) and report the mod of the K samples labels as the predicted label of the first test image. 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 classifying a new image (in terms of accuracy, runtime, …)? Why? (You may use the tic and toc commands of MATLAB to compare the calculation complexity of the methods) (4 points)**

Ans. The code for this can also be found above and in the attached file as well. I got an accuracy of 92.5% in this case which is definitely better than the previous case. This model takes a little bit more time to run when compared to the model in Question 5 (we can see in the image in the above question). The following is the output that MATLAB gave me for the accuracy and the time complexity of the model.

```
The accuracy of KNN model is 0.9250
Elapsed time is 0.177276 seconds.
```

As far as choosing the right model goes, I think it really depends on how large of a dataset do we have. However, in our case since we have a smaller dataset, I believe I will prefer KNN over the model in Question 5 as it is a more accurate. The is due to the reason that the model in Q5 only looks at closest value while KNN picks K closest values and then takes the value with the highest frequency and thus it is more reliable as compared to the model in Question 5.

Q7. Calculate eigenfaces weights matrix for your photo and compare it with $\boldsymbol{Warctichare}$ (arctic hare image weight matrix) and $\boldsymbol{Wmean}$ (training set weight matrix) that you had calculated before. What do you see? Treat your photo as a test image and test it on the implemented KNN algorithm. What similarities do you see between your photo and the predicted class image (Age, gender, glasses, …)? (3 points)

Ans. The code for this can be found above and in the attached file as well. The following is my photo that I used for testing:

After running the model, my face got matched with subject 34:

Just by looking at both the images we can notice that both the people are wearing glasses and the shape of the head is very similar. The age however is a completely

different story, and it does not match. Running the model by increasing the value of K (the number of eigen vector contributed), my face gets matched with subject 31.

Which indeed looks very similar to me as far as the hairstyle and the glasses go and has relatively less age when compared to the subject that I got matched to previously. This shows that selecting the right number of eigen vectors plays a key role to make the model more accurate.