ECE 442 Lab Report 2:

PCA and Eigenfaces

Section: H51

Compiled by: Abhi Sharma

ID: 1643951

NOTE

All the code can be found in the one single Lab2.mat file attached with the zip file.

1. PCA:

Code:

```
%Lab 1: Q1
%1.1

x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
y =  [2.56, 1.14, 4.01, 6.02, 4.62, 7.48, 7.79, 9.40, 10.43, 9.23, 13.22, 13.94, 14.30, 15.32, 1
m_x = mean(x)
m_y = mean(y)
x = arrayfun(@(x) x-m_x, x)
y = arrayfun(@(y) y-m_y, y)
plot(x,y)
xlabel("x")
ylabel("y")
title("x vs y")
cov_matrix = cov(x, y)
% In a covarience matrix, we know that the dioganal elements represent the
% varience of each feature and the off-diagonal elements represent the
% covarince of each feature
% In our case, since all the elemnets of the matrix are positive, all the
% features tend to increase or decrease together (i.e. if one feature
% increases so do the others, and if one feature decreases so does the
% other one


[e_vec, e_val] = eig(cov_matrix)

%e_vec is the matrix that represents the eigenvector
%and e_val is a matrix that represents the eigenvalues
```

```
[~, max_index] = max(diag(e_val)) % find the index of the maximum eigenvalue
max_eigenvec = e_vec(:, max_index) % extract the corresponding eigenvector


%%1.5



% Generate 100 random x and y coordinates between -10 and 10
rand_x = rand(1, 100)
rand_y = rand(1, 100)
m_rx = mean(rand_x)
m_ry = mean(rand_y)
r_x = arrayfun(@(rand_x) rand_x-m_rx, rand_x)
r_y = arrayfun(@(rand_y) rand_y-m_ry, rand_y)
% Plot the data points
scatter(r_x, r_y);
xlabel("x");
ylabel("y");
title("Centrized Random Data Points in 2D Space");
cov_mat_Q1 = cov(r_x, r_y)
[e_vec1, e_val1] = eig(cov(r_x, r_y))
```

Questions:

Q1. What assumption would you make about the dimensionality of the data? (1.6 points)

Ans. Before centering the dataset, it can be assumed that the data is a two-dimensional dataset with x representing the independent variable and y representing the dependent variable. After centering the dataset, the assumption that can be made is that the mean of the data is now equal to 0. Centering the data helps in bringing the data to the same scale and reduces the effect of large values in the data.

Q2. Extract the eigenvector with the highest eigenvalue (). Is there any relation between the direction of V and direction of the spread of the datapoints? (2.2 points)

Ans. The following is the eigen vector with the maximum eigenvalue:

```
max_eigenvec = 2×1
      0.6896
      0.7242
```

Yes, there is indeed a relationship between the direction of the vector V and the spread of the datapoints. The direction of V is represented by the principal component while the spread is spread is determined by the orientation of the principal component and hence, they are related.

Q3. Compute the covariance matrix, find the eigenvalues, and compare them. What would be the true dimensionality of the data? Can you prioritize one of the dimensions? (1.6 point)

Ans. The following are the covariance matrix, the eigen vectors and the eigen values:

```
cov_mat_Q1 = 2×2
10^6 ×
    8.6569   -0.0345
   -0.0345    7.9879

e_vec1 = 2×2
   -0.0514   -0.9987
   -0.9987    0.0514

e_val1 = 2×2
10^6 ×
    7.9861         0
         0    8.6586
```

The priority of one dimension over another can be determined by the magnitude of the corresponding eigenvalue. The dimension with the largest eigenvalue is considered to have the highest priority, as it captures the maximum amount of variation in the data. The other dimensions have a lower priority, as they capture less variation in the data.

In this case, the two eigenvalues are approximately equal, indicating that the two dimensions are roughly equally important in capturing the variation in the data. It is not possible to prioritize one dimension over another.

2. Eigenfaces

Code:

```
% Q2
folders = dir('Face\training');
folders = folders(~ismember({folders.name}, {'.', '..'}));
subFolders = folders([folders.isdir]);
mat = []
mean_mat = []
for K = 1:length(subFolders)
    cur_dr = ['Face\training\' subFolders(K).name];
    images = dir(cur_dr);
    images = images(~ismember({images.name},{'.','..'}));
    for i = 1:length(images)
        img = imread([cur_dr '\' images(i).name]);
        img = img(:, :, 1);        %turning the image into a greyside by only cosididering the f
        img = reshape(img, [], 1);
        mat = [mat img];
    end
end
mean_mat = mean(mat, 2);
size(mat)
whos("mat")
```

```
% Making our data X
im2 = reshape(mean_mat, [112, 92])
im3 = cast(im2, "uint8")
lol = size(im2)
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)


%Question 6 and Question 5


% First we calculate the eigenvalues and vectors for T with the dimensions
% 320*320
[evec_Q2, eval_Q2] = eig(T)
v_i = data_X*evec_Q2
max_6_eig_vec_total = [];
min_6_eig_vec_total = [];
[max_eig_val_Q2, num] = maxk(diag(eval_Q2), 6)
[min_eig_val_Q2, num_min] = mink(diag(eval_Q2), 6)
for i = 1:length(num)
    max_six_eigenvec = v_i(:, i);
    min_six_eigenvec = v_i(:, num_min(i));
    max_6_eig_vec_total = [max_6_eig_vec_total max_six_eigenvec];
    min_6_eig_vec_total = [min_6_eig_vec_total min_six_eigenvec];
end
```

```
temp_var = [];
for i = 1:6
    temp_var = reshape(max_6_eig_vec_total(:, i), [112, 92])
    %temp_var = cast(temp_var, "uint8")
    temp_var = (temp_var - min(temp_var, [], 'all')) / (max(temp_var, [], 'all') - min(temp_var, [], 'al
    imshow(temp_var)
    name_of_file_1 = sprintf("Q5_%dthEigen.bmp", i)
    imwrite(temp_var, name_of_file_1)
    temp_var_1 = reshape(min_6_eig_vec_total(:, i), [112, 92])
    temp_var_1 = (temp_var_1 - min(temp_var_1, [], 'all')) / (max(temp_var_1, [], 'all') - min(temp_var_
    %temp_var_1 = cast(temp_var_1, "uint8")
    imshow(temp_var_1)
    name_of_file_2 = sprintf("Q6_%dthEigen.bmp", i)
    imwrite(temp_var_1, name_of_file_2)
end
```

Questions:

Q4. Reshape the mean image to have the same dimension as each face image and plot it. Save mean image as 'mean.bmp' and attach it to the zip file.

(1.6 point)

Ans. Can be found in the code and the image can be found attached…

Q5. Pick the 6 eigenvectors (eigenfaces) with the largest eigenvalues as the new axes of the low-dimensional space. Reshape the 6 eigenfaces and plot them. Attach the resulted images "Q5_ithEigen.bmp" to the zip folder (3 points)

Ans. The images can be found attached to the zip file and the code can be found in the Lab2.mat file or the pictures above.

Q6. Plot the 6 eigenfaces with the lowest eigenvalues. Is there any information in the plots? Do they look like a generic face image? Attach the resulted images "Q6_ithEigen.bmp" to the zip folder (2.4 points)

Ans. The eigenfaces in question 6 do look a little bit like eigenfaces with the small details of the face present while the contrast missing. We know that the eigenfaces with the highest eigenvalues carry the information about the contrast of the image while the eigenfaces with the lowest eigenvalues contain the information about the sharpness and the detail of the image. Looking at the figures one can clearly observe that this is indeed correct. The eigenfaces with high eigen values give more detail about the contrast of the image and the images are darker while the eigenfaces with the low eigenvalues contain finer details about the facial features of the person while the contrast and the darkness are completely missing.

The code and the images can be found attached.

3. Eigenfaces extraction using SVD decomposition:

Code:

```
%Question 7
[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');
for i = 1:6
    eig_6_max_vec_SVD = eig_faces_SVD(:, eig_val_indices(i))
    eig_6_max_vec_SVD = reshape(eig_6_max_vec_SVD, [112, 92])
    name_of_file_3 = sprintf("Q7_%dthEigen.bmp", i)
    %imshow(eig_6_max_vec_SVD, [])
    eig_6_max_vec_SVD = (eig_6_max_vec_SVD - min(eig_6_max_vec_SVD, [], 'all')) / (max(eig_6_max_vec_SVD
    imwrite(eig_6_max_vec_SVD, name_of_file_3)
end
```

Q7. Plot the 6 eigenfaces with the highest eigenvalues again (supposed to see the same results as Question 5). Attach the resulted images "Q7_ithEigenSVD.bmp" to the zip folder. (2 points)

Ans. The code and the images can be found attached in the zip file….

4. Reconstruction:

Code:

```
%Q8:
lol1 = size(U)
eig_sixty_face = U(:, 1:60);
eig_sixty_face_t = eig_sixty_face'
image_mat = imread("Face\training\s2\1.png")
imwrite(image_mat, "Q8org.bmp")
image_mat = image_mat(:, :, 1)
image_mat = reshape(image_mat, [], 1)
image_mat = cast(image_mat, "double")
W = eig_sixty_face_t*(image_mat-mean_mat)
%Reconstruction using 60 eigenfaces
reconstructed_mat = (eig_sixty_face * W) + mean_mat
reconstructed_mat_img = reshape(reconstructed_mat, [112, 92])
reconstructed_mat_img = (reconstructed_mat_img - min(reconstructed_mat_img, [], 'all')) / (max(reconstru
imshow(reconstructed_mat_img)
imwrite(reconstructed_mat_img, "Q8rec.bmp")
%Q9
%reconstruction using 120 eigenfaces
eig_120_face = U(:, 1:120)
eig_120_face_t = eig_120_face'
W_120 = eig_120_face_t*(image_mat-mean_mat)
reconstructed_mat_120 = (eig_120_face * W_120) + mean_mat
reconstructed_mat_img_120 = reshape(reconstructed_mat_120, [112, 92])
reconstructed_mat_img_120 = (reconstructed_mat_img_120 - min(reconstructed_mat_img_120, [], 'all')) / (ma
imshow(reconstructed_mat_img_120)
imwrite(reconstructed_mat_img_120,"Q9rec.bmp")
```

Q8. Attach the original image and its reconstruction named as 'Q8org.bmp' and 'Q8rec.bmp', respectively to the zip file. (3.6 point)

Ans. The code can be found above (also in the Lab2.mlx), and the images are attached with the zip file.

Q9. Now choose 120 eigenfaces. Reconstruct the original image again.

and plot it. What do you see? How would you choose the optimal number of

eigenfaces? Save and attach the result of reconstruction (Q9rec.bmp) to the zip file. (2 points)

Ans. When we use more eigenfaces to plot the image, we can see that there is more clarity in the image, and it is a bit more detailed.

The goal is to retain as much information as possible while reducing the dimensionality of the data. The optimal number of eigenfaces is therefore a trade-off between information retention and computational storage. Hence, while choosing the number of eigenfaces we need to maintain a balance between the computational aspects and the quality of the image that we need to maintain.