AMATH 582 HW4: Eigenfaces and Music Recognition

Huasong Zhang
Department of Applied Mathematics
University of Washington
Seattle, WA 98195
`huasongz@uw.edu`

Abstract

In this assignment, there are two parts. The first part is about eigenfaces, where we can do analysis on the training set of face images and do classifications or reconstructions on test images. The second part is also related to classification. However, this time, we are dealing with music clips. We intend to do svd on the spectrogram of music clips and then apply classification methods.

1 Introduction and Overview

Nowadays, classification has been widely used in our daily life. We often see that on Facebook page, when you pose a photo with someone, it will automatically match the face with someone that might be. This is one of the use of face recognition. Of course there are many other uses of face recognition. So in this assignment, we are going to investigate on how singular value decomposition applies on face images and reconstruct the faces.

For part two, the idea is similar. There are many genre of music in this world. The frequency, pitch, timbre, and other measurements are different for each genre. Our human beings can easily tell the difference between genres; however, we want to train machines so that they can separate and classify the music clips for us. In part two, we will do three tests. The first one we have three artists and each of them has their style of music. For the second test, all music clips are from Seattle bands but we have three bands here. So the music styles are similar for them. In test three, we have three different genre of music, but the clips are from different artist. The task for Part two is to classify within each test.

2 Theoretical Background

Since the training and testing set of images and musics are large, singular value decomposition is important to decompose the high dimensional matrix into low dimensional. Take the image matrix as an example, the dimension of it will be m by n, where m is the time and n denotes number of images. Any matrix X can be written in the form:

$$X = U\Sigma V^* \tag{1}$$

This is singular value decompostion(SVD). U is a m by n matrix where each column represents the axis for the new basis and U is the feature space. Σ is the singular value matrix with dimension n by n. It is a diagonal matrix and the values on the diagonal are in the descending order, where the first one has the largest value and it means that coordinate is the most important one. And the number of significant singular values is the rank r. Instead of using the full rank to reconstruct the faces, we can use a small value r and it will also gives us pretty good results. The last matrix V has dimension n by n as well. It is a matrix which represents the signature of the faces.

To reconstruct the image, we can choose a small number r based on the energy we want. Higher the energy, larger r, and the reconstructed images will be more clear. To reconstruct, we use:

$$reconstruct = U * \Sigma(:, 1:r) * V'(:, 1:r) \tag{2}$$

on all the images.

For second part of this homework, we need to classify the music. There are many classification methods. Here, I used the K-nearest neighbor method(KNN) and support vector machine(SVM).

KNN works by choosing one data and observing the nearest k neighbors. Then the class of this data will be classified as the same as the majority of the data in nearest k neighbors. I choose this method because it's easy and efficient.

Another ways is SVM. When we want to separate two classes, there mayby many ways to draw the decision line. SVM aims to find the line where the distance between the decision line and the closest data point can be maximized on both sides.

3 Algorithm Implementation and Development

3.1 Part I

For face images, the first thing is to read the image, convert all the images into column vector and save them together. Then I did SVD on the high dimensional image matrix and got U, σ, and V. I divided the numbers on the diagonal of Σ by the sum and plotted it. The number of dots that stand out will the the rank r, so that I know the first r components contributed the most. Then I build the reconstruction matrix in order to reconstructed faces. I did the above for both cropped faces and original faces. I plotted singular values for both cases and reconstructed some sample faces. It can be seen that there are some differences between them.

3.2 Part II

For music, I also need to do pre-processing. For each 5-second music clips, there are two columns, which is L channel and R channel. I took the mean of these two columns and use only one column to represent one music clip. I have 90 music clips in total for each test. After getting music matrix, I randomly generate 10 numbers and took out these column vectors as my test set and train on the rest of columns. Instead of directly doing SVD on the music matrix, we need to do SVD on spectrogram since spectrogram is in frequency space and that is where the different genre of music differs from each other. For both training and testing set, I had a for loop in order to collect the sepctrogram. Within the for loop, I took out each column, did fft on it, and saved the absolute value of fftshifted columns in the spectrogram matrix. After getting the spectrogram for all the training music clips, I did SVD and get U, Σ, and V matrix. Since V is the signature matrix of songs, I put V and the labels for my training set into KNN to train my data. Last, I used the trained KNN and SVM model to predict the class for my testing set and calculated the accuracy.

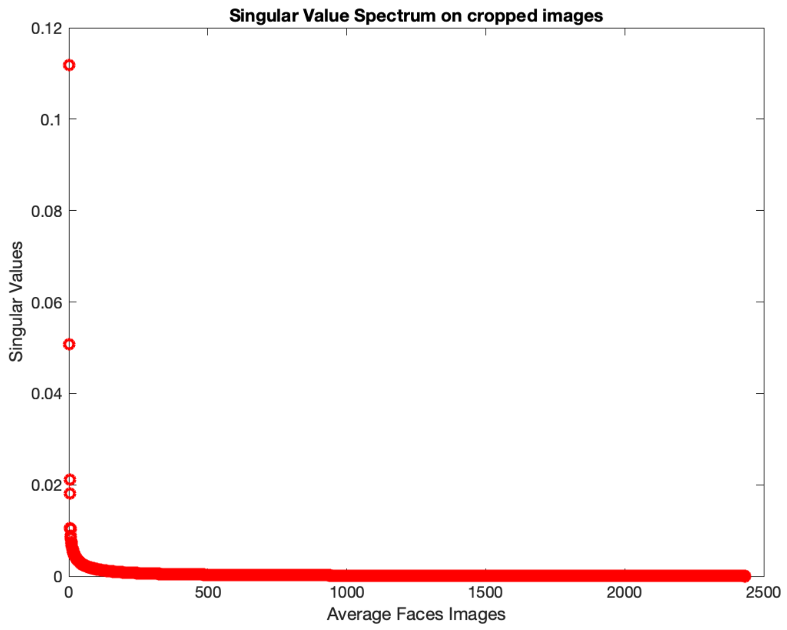
4 Computational Results

4.1 Part I

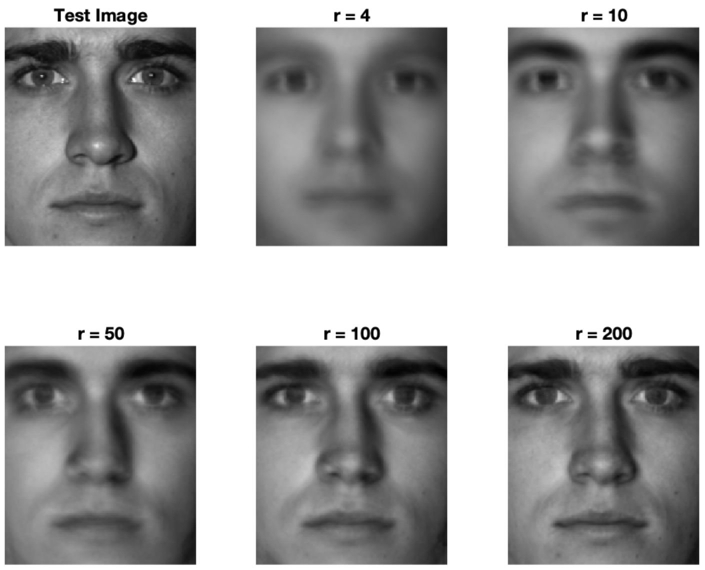
As mentioned before, we decomposed the image matrix x into U, Σ, and V. U is the feature space. It is the new space where the images are projected on. Σ is a diagonal matrix where the values on the diagonal are in descending order. The larger the value, the more important the corresponding coordinate in U would be. And V is the signature matrix of images. It shows what the feature is on each coordinate or axis in the new space.

In terms of interpreting plots, the following figure shows the singular values of the cropped faces. From this plots we can see that the fir 4 are the most important. Therefore, it has rank $r =$

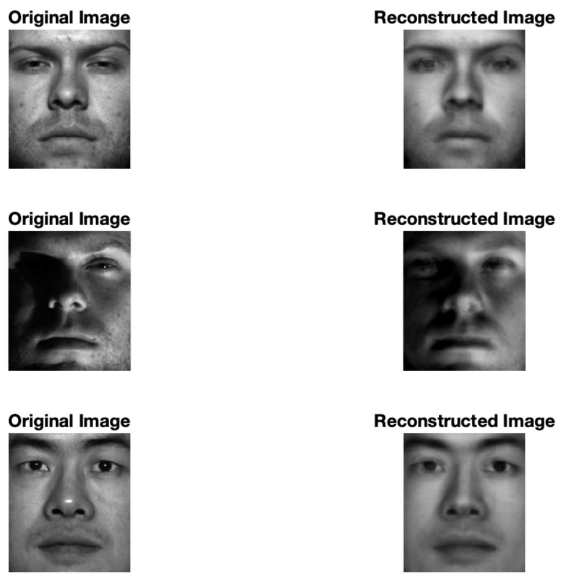
4. It means only using the first 4 columns, we can get pretty good results on reconstructed matrix.



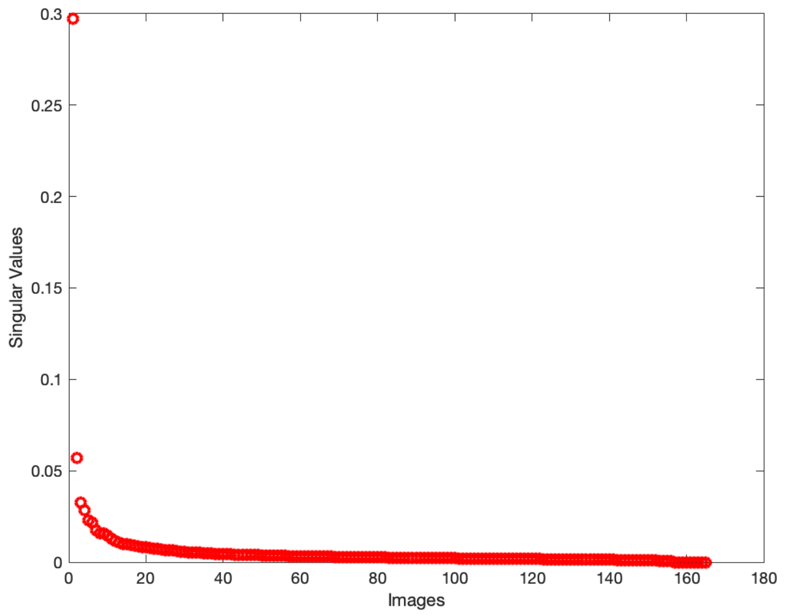
And the results are shown in the figure below where I tried different rank numbers on reconstructing the same image. As we can see, when $r = 4$, we can already see a blurry face. When the r increases, the face becomes more and more clear. So for rough estimates, rank 4 would be enough.



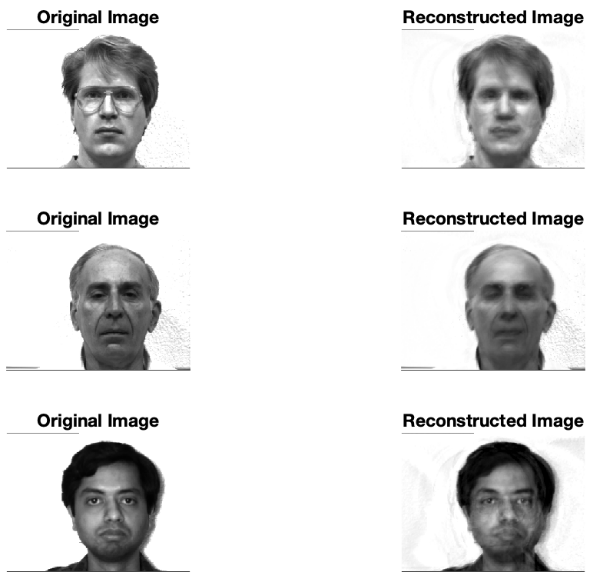
In the figure below, I used the same number of for the rank and constructed different images. In this case, I chose $r = 50$. As shown in the picture, the results is pretty good no matter how the light is in the image. Even for the second one when there is a large area covered under the shadow, the reconstructed image is still pretty clear.



In the following, the singular values for original faces are shown here. It can be seen that the first two are far away from the others, especially the first one. Itself takes up 30 percent of the energy.



In reconstructed orignial faces as shown in the following figure, I used the same number for the rank as I chose for cropped. However, as we can see from the plots that it is not as clear as the cropped. This maybe caused by the surroundings and background in the images.



4.2 Part II

The results for applying KNN classification method on three tests are:

```
Accuracy for test 1 using KNN is 0.6
Accuracy for test 1 using SVM is 0.7
Accuracy for test 2 using KNN is 0.2
Accuracy for test 2 using SVM is 0.1
Accuracy for test 3 using KNN is 0.7
Accuracy for test 3 using SVM is 0.9
```

As we can see from the results that test 1 and test 3 have high accuracy, but the accuracy for test 2 is pretty low. It actualy makes sense. For test 1 and 3, the style or genre of music for three classes are very different. Therefore, their spectrograms in frequency domain will be distinct. When we train the data and predict on test set, the decision line would be pretty clear. However for test two, we have the same style of the music just artists are different. Even though we have different artists, the main style will be the same; there will be only small differences between them three. So it is hard for machine to tell which music clip comes from which artist, not even mention the music clip is only 5 seconds for each. It is difficult to do classification on test 2. Therefore, it is expected to have the lowest accuracy.

5 Summary and Conclusions

5.1 Part I

By doing this assignment on SVD we can clearly see that SVD can efficiently change a high dimensional matrix into low dimensional. In that case, it saves time and space; it is easier, faster, and more convenient to do things on the new basis but also focusing on the important information containing in the matrix. Therefore, it is concluded that SVD is very useful in data analysis, especially in data mining and cleaning.

5.2 Part II

In Part two, I used KNN and SVM as the classification method. KNN is easy to understand and easy to implement. And as we can see from the results that the accuracy is pretty high for classes have big differences. So this method is efficient in some way. SVM is stronger than KNN. There are many other methods that we can use, such as naive Bayes, random forest and many other more. It seems like SVM gives me higher accuracy. Since my dataset is not large enough, it is hard to say which method is really better than the other one. The high accuracy may occur just by chance.I think different dataset may has different best way to classify.

Appendix A MATLAB functions used and brief implementation explanation

[u,s,v] = svd(x)
This command will do the singular value decompose on matrix x, and return three matrices.

knn.mod = fitcknn(X1,Y1,'NumNeighbors',5)
This command will train the K-nearest neighbot model on the training set. X1 is the dataset and Y1 is the correspoding labels for the training set. The number of neighbors can be adjusted based on real life situations.

svm.mod = fitcecoc(X1,Y1)
This command is the multiclass svm training command. X1 is the training data and Y1 is the corresponding labels for training data.

label = predict(model,X2)
This command will predict the labels for the test set using the model that trained before. Here, X2 is the testing set data, and the output will be the corresponding predicted test labels.

Appendix B MATLAB codes

Part I

```
clear all;clc;

% load files
mainFolder1 = dir('CroppedYale');
mainFolder1 = mainFolder1(4:41);
mainFolder2 = dir('yalefaces_uncropped');
mainFolder2 = mainFolder2(4);

%% Cropped
ave_face = [];
CROP = [];
for i=1:length(mainFolder1)
    name = mainFolder1(i).name;
    a = ['CroppedYale','/',name];
    subfolder=dir(a);
    subfolder = subfolder(3:end);
    Cropped = [];
    for j = 1:length(subfolder)
        data = imread(subfolder(j).name);
        data = reshape(data,192*168,1);
        Cropped = [Cropped data];
        CROP = [CROP data];
    end
    ave_face = [ave_face sum(Cropped,2)/length(Cropped(1,:))];
end
% CROP: all the images, each column represents an image
% ave_face: each column is the average face for each person

% SVD
CROP = double(CROP);
[U,S,V] = svd(CROP,'econ');
[U1,S1,V1] = svd(ave_face,'econ');

% Plot
figure(1)
subplot(1,2,1)
```

```
plot(diag(S)/sum(diag(S)),'ro','LineWidth',[2])
ylabel('Singular Values')
xlabel('Images')
title('Singular Value Spectrum on all cropped images')
subplot(1,2,2)
plot(diag(S1)/sum(diag(S1)),'ro','LineWidth',[2])
ylabel('Singular Values')
xlabel('Average Faces Images')
title('Singular Value Spectrum on average cropped images')
print(gcf,'-dpng','crop_singular_value_spectrum.png');
% The plot shows it has rank 4
%%
reconstruct = U*S(:,1:50)*V(:,1:50)';

img = [2,50,100];
figure(2)
for i = 1:length(img)
    subplot(3,2,2*i-1)
    imshow(uint8(reshape(CROP(:,img(i)),192,168)));
    title('Original Image')
    subplot(3,2,2*i)
    imshow(uint8(reshape(reconstruct(:,img(i)),192,168)));
    title('Reconstructed Image')
end
print(gcf,'-dpng','Reconstruct_cropped_image.png');


test = imread('yaleB33_P00A+010E+00.pgm');
figure(3)
subplot(2,3,1)
imshow(test)
title('Test Image')
test = double(reshape(test,192*168,1));
% rank 4
rank = [4,10,50,100,200];

for i = 1:length(rank)
    U_new = U(:,1:rank(i));
    recon = U_new*U_new'*test;
    recon = reshape(recon,192,168);
    subplot(2,3,i+1)
    imshow(uint8(recon))
    t = ['r = ',num2str(rank(i))];
    title(t)
end
print(gcf,'-dpng','Reconstruct_cropped_image1.png');

%% Original
%ave_face_original = [];
ORIGINAL = [];
for i=1:length(mainFolder2)
    name = mainFolder2(i).name;
    a = ['yalefaces_uncropped','/',name];
    subfolder=dir(a);
    subfolder = subfolder(3:end);
    original = [];
    for j = 1:length(subfolder)
        data = imread(subfolder(j).name);
        data = reshape(data,243*320,1);
```

```
          original = [original data];
          ORIGINAL = [ORIGINAL data];
     end
     %ave_face_original = [ave_face_original sum(original,2)/length(original(1,:))];
end
% CROP: all the images, each column represents an image
% ave_face: each column is the average face for each person

% SVD
ORIGINAL = double(ORIGINAL);
[U2,S2,V2] = svd(ORIGINAL,'econ');
%[U3,S3,V3] = svd(ave_face_original,'econ');

% Plot
figure(4)
plot(diag(S2)/sum(diag(S2)),'ro','LineWidth',[2])
ylabel('Singular Values')
xlabel('Images')
print(gcf,'-dpng','original_singular_value_spectrum.png');
% The plot shows it has rank 2

% Test
[test,B]= imread('subject03.normal');
%figure(4)
%subplot(2,3,1)
%imshow(test)
%title('Test Image')
%test = double(reshape(test,243*320,1));
% rank 4
reconstruct = U2*S2(:,1:50)*V2(:,1:50)';

img = [2,50,100];
figure(5)
for i = 1:length(img)
     subplot(3,2,2*i-1)
     imshow(uint8(reshape(ORIGINAL(:,img(i)),243,320)));
     title('Original Image')
     subplot(3,2,2*i)
     imshow(uint8(reshape(reconstruct(:,img(i)),243,320)));
     title('Reconstructed Image')
end
print(gcf,'-dpng','Reconstruct_original_image.png');
```

Part II

Only codes for Test 3 is listing here. Code for Test 1 and Test 2 are the same.

```
clear all;clc;

mainFolder3 = dir('Test 3');
mainFolder3 = mainFolder3(4:end);

y = ones(1,30);
Y = [y 2*y 3*y];
y3 = Y;

% Test 2
% Load Test 2 data
y_3 = zeros(240000,1);
Fs_3 = [];
```

```
n = 240000;
for i=1:length(mainFolder3)
    name = mainFolder3(i).name;
    a = ['Test 3','/',name];
    subfolder=dir(a);
    subfolder = subfolder(3:end);
    for j = 1:length(subfolder)
        [y,Fs] = audioread(subfolder(j).name);
        y = mean(y,2);
        if length(y(:,1))>n
            y = y(1:n,:);
            %Fs = Fs(1:n,:);
        else
            n = length(y(:,1));
            y_3 = y_3(1:n,:);
            %Fs_1 = Fs_1(1:n,:);
        end
        y_3 = [y_3 y];
        Fs_3 = [Fs_3 Fs];
    end
end
y_3 = y_3(:,2:end);
test3 = [];
for j = 1:2:length(y_3(1,:))-1
    b = reshape(y_3(:,j:j+1),length(y_3(:,1))*2,1);
    test3 = [test3 b];
end
%y_3 = y_3(:,2:end);
test3 = y_3;

% Separate into training and testing set
ind = randi([1 90],1,20);
test3_test = test3(:,ind);
y3_test = y3(:,ind);
test3(:,ind) = [];
y3(:,ind) = [];
test3_train = test3;
y3_train = y3;

% Spectrogram
spec_train = [];
for i = 1:length(test3_train(1,:))
    ft = fft(test3_train(:,i));
    spec = abs(fftshift(ft));
    spec_train = [spec_train spec];
end

spec_test = [];
for i = 1:length(test3_test(1,:))
    ft = fft(test3_test(:,i));
    spec = abs(fftshift(ft));
    spec_test = [spec_test spec];
end

[a,b]=size(spec_train); % compute data size
ab=mean(spec_train,2); % compute mean for each row
spec_train=spec_train-repmat(ab,1,b); % subtract mean

[c,d]=size(spec_test); % compute data size
```

```
cd=mean(spec_test,2); % compute mean for each row
spec_test=spec_test-repmat(cd,1,d); % subtract mean

% SVD
[U3,S3,V3] = svd(spec_train','econ');
figure(3)
plot(diag(S3)/sum(diag(S3)),'ro','LineWidth',[2])
xlabel('Song clips spectrogram')
ylabel('Singular values')
title('Singular value spectrum for Test 3')
print(gcf,'-dpng','test3_singular_value_spectrum.png');
% rank = 10

% KNN
knn.mod = fitcknn(V3',y3_train','NumNeighbors',5);
label = predict(knn.mod,test3_test');
right = 0;
for i = 1:length(label)
    if label(i) == y3_test(i)
        right = right + 1;
    end
end
accuracy = right/10;
A = ['Accuracy for test 3 using KNN is ', num2str(accuracy)];
disp(A)

% SVM
svm.mod = fitcecoc(V3',y3_train');
label = predict(svm.mod,test3_test');
right = 0;
for i = 1:length(label)
    if label(i) == y3_test(i)
        right = right + 1;
    end
end
accuracy = right/10;
A = ['Accuracy for test 3 using SVM is ', num2str(accuracy)];
disp(A)
```