AMATH 482 Homework 4

Section 1. Introduction and Overview

In this assignment, we continue to use Principal Component Analysis to analyze a large collection of data through their Principal Components (PC). We extended the PCA analysis by enhacing it with the concept of supervised learning (Naïve Bayes Fit) technique to classify the projection of each data element to classify its group. The PCA analysis is explored in section 1 of this report through processing face images, while the classification and machine learning is explored in section 2 of this report in which we train a Naïve Bayes Model to classify 5 second music clips.

Section 2. Theoretical Background

Principal Component Analysis is a method to reconstruct the our data into the unitary basis transformation matrix **U**, the singular values corresponding to the variance of each principal components **Σ**, and the unitary principal component bases. We can see in the results section that **Σ** is used to plot the singular value spectrums, which is used to determine the dominant components. **V** represents the unitary principal bases, which can be used to reconstruct the reduced rank data, which is useful for ompression. In addition, we also use the principal component projections **U.'*X** where **X** is the data we pass to the SVD function. This will represent **X** in the principal component direction, which is useful to see the principal or distinguishing features of the matrix **X**. Furthermore, we can project **U.'*X** to our data matrix **X** to obtain the score of each row of the matrix to its Principal Components, which can be used as a feature to classify the label corresponding to the row projected.

In this assignment, we are still using the idea of Gabor Transform to produce a spectrogram. This will allow us to understand not only the available frequencies in a data, but also the time when those frequencies occur, which can be used to distinguish signals such as music clips in this example. This process involves using the FFT command, which assumes the sampling domain range of 2*pi, which means that if we want to recover the appropriate time stamp, we need to rescale it to 2*pi/L.

Lastly, the model we used: Naïve Bayes Classifier works by creating probability maps for the points we provided (the label classifiers). It is a supervised learning algorithm which requires us to provide the label to each of the data set we pass. It computes the probability that the data belongs to a label by multiplying all the constructed possibilities that is created based on the parameters or dimensions or in our assignment's case: the scores (length of projection on a principal component) corresponding to each label. If the probability of the data to be predicted is higher than the probability of the model that leads to the option chosen, the model will predict its corresponding label, and vice versa, will predict other options if the probability to the current label is less than the threshold.

Section 3. Algorithm Implementation and Development

YaleFaces

Before we get to the PCA analysis, we need to load all the images to our workspace with the correct naming convention. I implemented the dir() command which reads all the files in the directory, where I opened each file and used imread() to load all the images into a uniquely named matrix. This method is implemented to load the images in the cropped and uncropped folders. Once all the files have been loaded, I saved the workspace as "loaded.mat", which avoids the need to reread the files.

I continued the analysis by preparing the X matrix for the cropped pictures. The X matrix contains all the reshaped images which we will analyze with PCA. Each image is flattened to a vector, and the image vector is

stacked in a row matrix for consistency with the previous assignment and the next part of the assignment (although we are instructed to store it in each column). Once X is ready and converted into double, we can start performing SVD which yields matrix U (unitary bases transform matrix), S (covariance matrix), and V(the new principal bases). Lastly, the principal component projection of the original data is projected to the principal components by doing U.'*X, which is stored as Y.

We can then take the diagonals of S and plot it to observe the modes. We can also plot each row of Y (containing the principal projection imges) and X (containing the original images). Furthermore, we can perform rank reduction where the new rank is decided from the number of PCA modes (energy plot) that are non-zeros and relatively big. After only accessing the first until the r^{th} element of U, S, and V, we can reconstruct X from the reduced rank SVD coefficients (stored as Xr). We can then plot each row of Xr and compare it with X to see the effect of rank reduction.

Music Classification

Similar to YaleFaces, we also need to load all the music to the workspace using the dir() command. For each test, 3 groups of 5 music clips are used for the analysis. Since the files loaded are music files, we need to perform audioread() to each file, which will return a stereo matrix of the song, where the elements are the amplitude at each timestep. Therefore, I averaged out the two amplitude streams to get a mono matrix of the music. The other prerequisite is to find out the sampling frequency of the clip, which is returned when calling audioplayer() (Fs = 44000 in my case). Figuring out Fs helps us check our file naming convention by playing them with playblocking(), and to figure out the timestamp of the clip in seconds which is useful to create the proper spectrogram. Moreover, I downsampled the song by only taking every other 8^{th} data point in order to make the music clips smaller in size to decrease the computing load. Each music clip is then sampled 20 times, where 5 second clips are cropped from each songs. These clips will be used to create the training data.

To create the training data, we need to present our clips in a highly distinguishable form. Hence, we can use Gabor Filtering to create the spectrogram, which is then rearranged in its principal component projection form by using PCA. The spectrogram serves to distinguish the beat of the music and the pitches and overtunes identified in a clip, which hopefully enhances the classification of each music. To obtain the spectrogram, after the fourier coefficients are prepared, we perform Gabor filtering by determining the time window and the t_{slide} which are good enough to create a distinguishable spectrogram. The adjustment is done by analyzing one music clip and plotting it's spectrogram to see the profile and matching it with the sound of each clip. After both parameter is better adjusted, we can perform Gabor filtering for all the music clips, and we can append each reshaped spectrogram in the big X matrix which will be analyzed using SVD.

Furthermore, the we used SVD to perform PCA analysis which presents our music clips in just the dominant principal components determined from their energy plots to decrease computational load while having the best coordinate system. We can obtain the principal component projection by taking the dot product of U and our matrix containing the collection of spectrograms (U.'*X). We can observe the energy plots, and determine the relatively large components, whose components will be used as the training data. After the dominant components are determined, we can project the spectrogram data matrix (X) to the principal component projection Y to obtain the projection of each music to the first few principal component directions to analyze the scores of each music clip with its principal component.

The scores collected is used as the training data (Xtrain) of the Naïve Bayes model. We can visualize the scores by plotting Xtrain on a 3D plot, where we can observe how our music is clustered in which we can obtain information about how well the classification algorithm can perform. After preparing the label that corresponds

to the group name of each row's scores, we can run Xtrain and its label to fitcnb() to create the model used to predict other music clips.

We can then take new music clips from the same musics we have not sampled yet, and we can prepare the spectrogram, and project it to the principal components just like how we processed the training data. Once the scores containing the clips to be predicted is obtained, we can run it under predict() by passing our previous model, and this will make predictions of the classification of each music clip.

In my case, I collected 15 clip samples from each music group, and I ran them to the predict() function. I then calculate the accuracy of the prediction by taking the ratio of the correct guesses to the total samples (45). The model is helpful if the accuracy of prediction is higher than random (33.3%) since there are three groups to predict. These procedure is repeated for each test case.

Section 4. Computational Results

Yale Faces

The figures below describes the principal components of the culmination of the faces (U.'*X). We confirm that the **U** matrix consists of the unitary basis transformation to the principal components from the principal faces. Therefore, the principal faces characterizes the dominant facial features among all the pictures. The energy plot described in the figure below represents the elements of the diagonal elements in the matrix **Σ,** which is the variance of each principal face. We can take the principal components corresponding to the high variance to be considered as the dominant component. In the end, I was able to rank reduce the **U**, **Σ, V** matrix in order to reconstruct the original data with just the consideration of the principal components.

Cropped Faces

The principal faces in Figure 1. show the dominant distinguishing facial features such as the eyes (1st component), the nose (3rd component), the mouth (4th component), and so on. I took the first 4 components to be considered as the dominant components based on the energy plot.

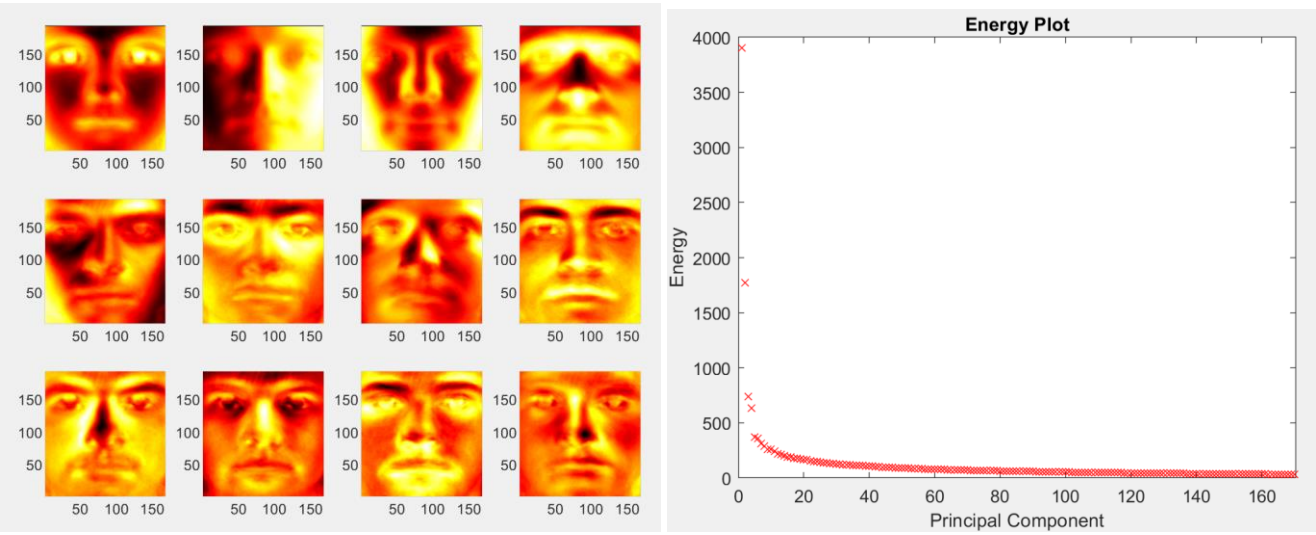

Figure 1. Principal Faces and the Singular Value Spectrum

Figure 2. below shows the variety of the faces provided and its corresponding reduced rank reconstruction. The singular value spectrum was not really helpful for me to determine the treshold rank reduction value. Hence, I experimented with the rank values and concluded that the first 80 principle components are needed to reconstruct the image. This is great since we have reduced the rank to 3%.

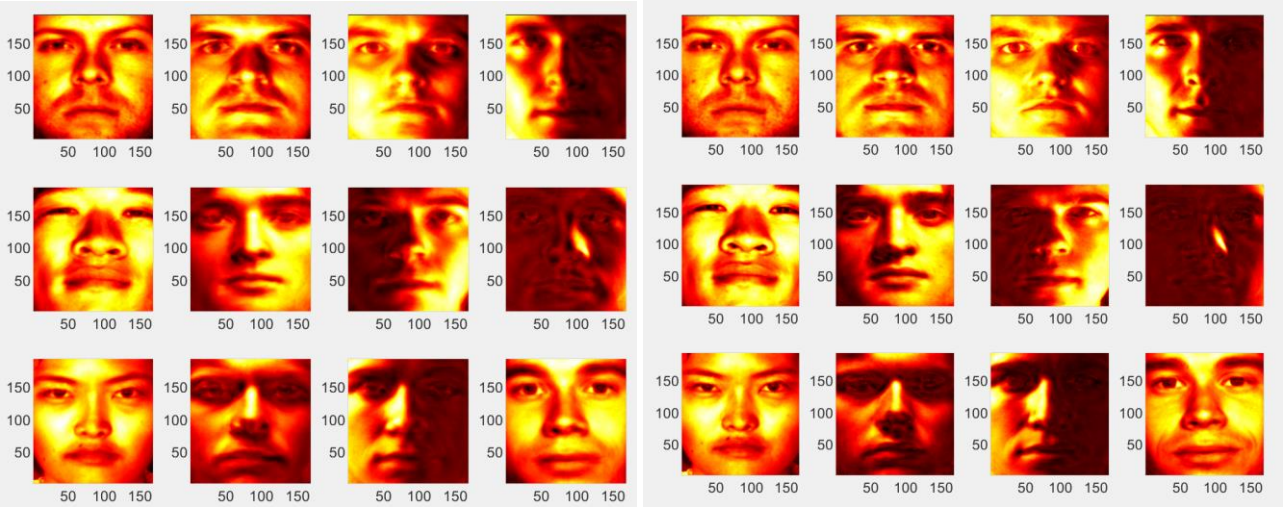

Figure 2. Faces Provided and the Reconstructed Reduced Rank Faces

Uncropped

The uncropped picture's principal components shown in Figure 3. are harder to be interpreted as the dominant facial features. One possible reason is because the position of the dominant features changes from picture to picture, which results in the difficulty to identify distinct features. We can see that the variance magnitudes are not as high as those of the cropped images, which means that the principal directions are not as dominant.

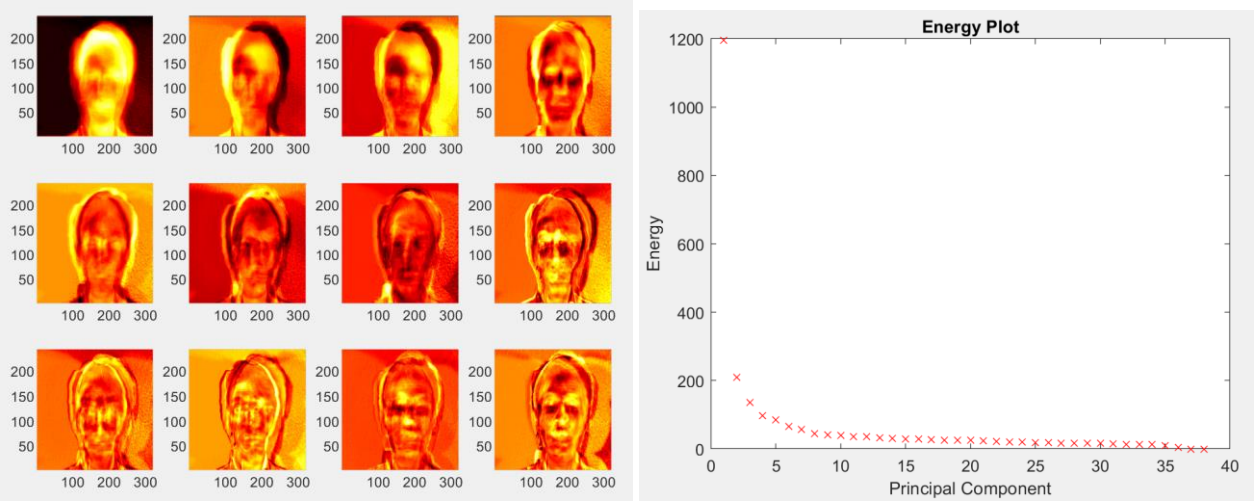

Figure 3. Principal Faces and the Singular Value Spectrum

I decided to reduce the rank of the SVD matrices to 32, which results in the pictures shown in Figure 4. The reconstruction of the faces resembles the original pictures. We only reduced the rank to 84.2%, which is not great at all, and this is caused by the noise introduced by having the characteristic features of the face being located at different places.

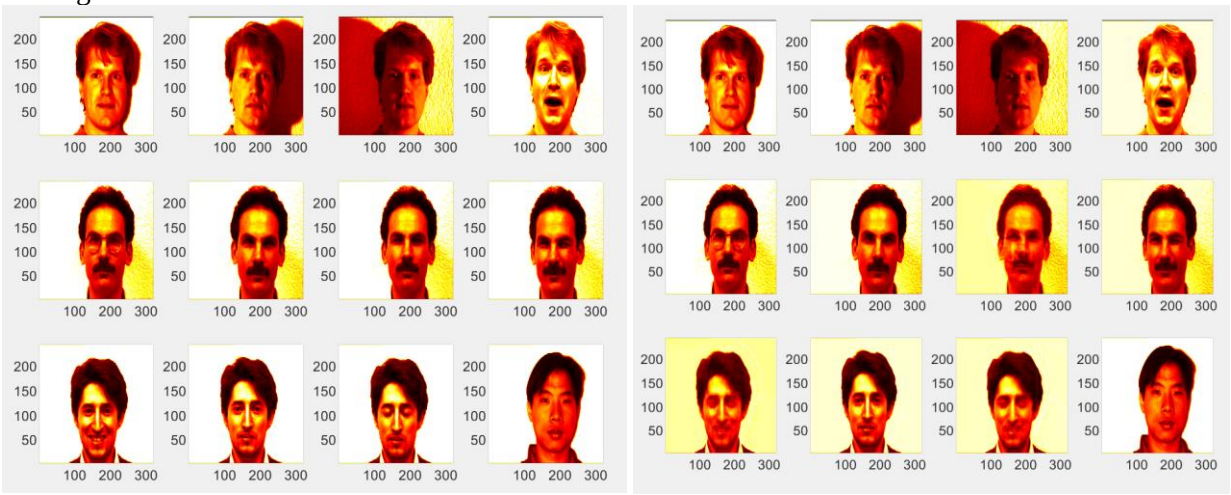

Figure 4. Original Images and the Reconstructed Reduced Rank Faces

(test 1) Artist Classification:

Accuracy: 60 percent

For test 1, I sampled three groups of music performed by different artists: Notorious B.I.G., Chrisye (Indonesian Singer), and ACDC. The resulting scores plot of each group is shown in Figure 5. below, where the thinner spots are the training sets and the thicker spots are the ones to be predicted. **The accuracy of the trained model is 60 percent** with 300 samples, which is much higher than random (33 percent). We can see that the scores have distinct clusters which makes them easier to be distinguished. For every tests, I obtained the scores by projecting the datasets to the 4 first principal components. The energy plots shown alongside the scores plot shows how dominant each components are.

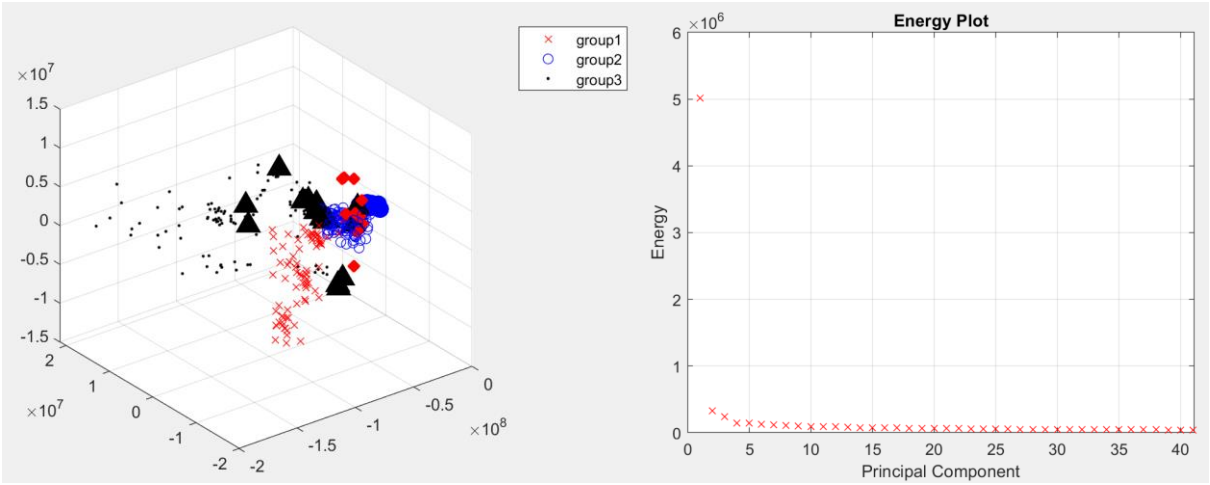

Figure 5. Scores Plot and the Singular Value Spectrum of Test 1

(test 2) Same Genre Different Artist Classification:

For test 2, I classified three bands from the same genre: Pink Floyd, Queen, and Yes. The resulting scores plot shown in Figure 6. Shows how closely clustered all the groups are which makes it hard to classify. In addition, there are a lot of dominant components in the singular values, which means that it is required to use more principal components, which I did not do to train my model. **As a result, the accuracy of the prediction drops to as low as 42.2 percent.**

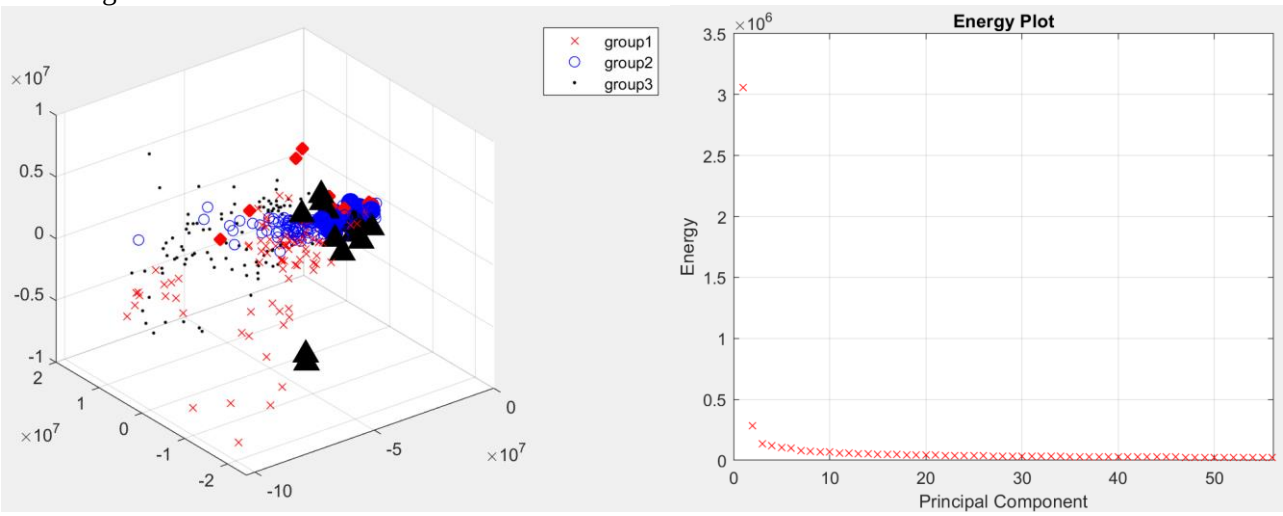

Figure 5. Scores Plot and the Singular Value Spectrum of Test 2

(test 3) Genre Classification:

Lastly, for test 3, I chose musics from 3 different genre: pop, classic rock, and 8-bit music. **The accuracy of this classification is 51.1 percent**, as we can tell that the singular value spectrum is relatively high and the clustering of the scores are a little more apparent.

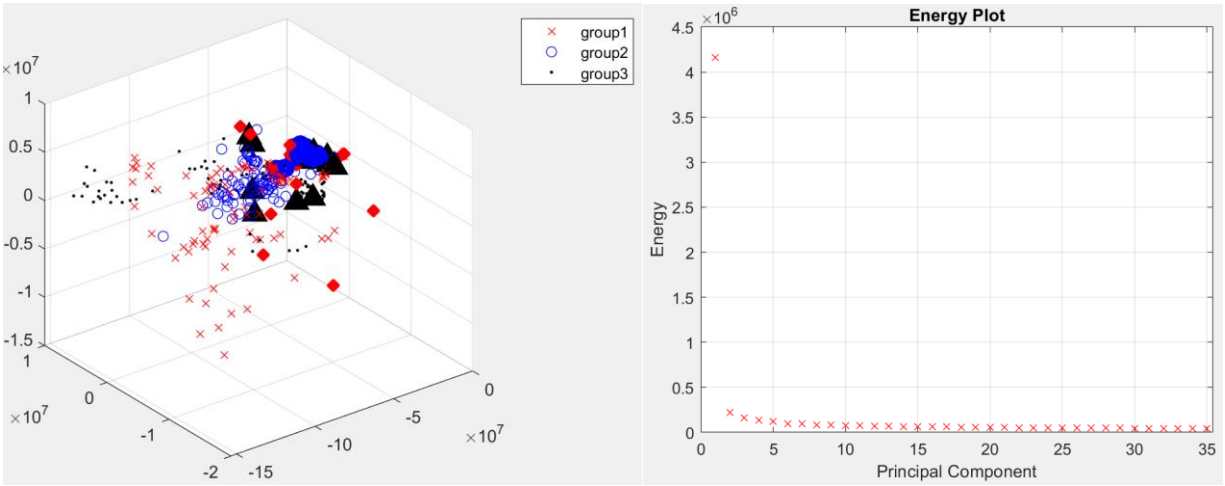

Figure 6. Scores Plot and the Singular Value Spectrum of Test 3

Section 5. Summary and Conclusion

In conclusion, PCA analysis is extremely useful to perform compression of data, where we can just use the dominant principal components and greatly reduce the rank of our data matrix and the number of ranks needed can be deduced from the singular value spectra. This advantage holds especially when the data we have are not noisy and consistent. With the idea of rank reduction, we can perform machine learning based on the reduced rank data component to perform quicker and simpler computation. We used Naïve Bayes Classifier model to classify our data, and based on my results, I need more data sets in order to accurately predict the label of a data set, since the maximum accuracy I achieved is 60%.

Appendix A: MATLAB Functions Implementation Explanation

`files = dir(*.*)`

Returns a file object containing all the files or directories inside the current path.

`image = imread(A)`

Returns the matrix representation of an image.

`audio = audioread(A)`

Returns the matrix representation of the audio object.

`eval([statement])`

Executes the char array specified to the workspace.

`num2str(number)`

Returns the string type of the number entered.

`[U S V] = svd(X, 'econ')`

Returns the SVD matrices of X using computationally cheap algorithm.

`lambda = diag(X)`

Returns the diagonal elements of matrix X.

`mdl = fitcnb(Xtrain, label)`

Returns the model object used to make predictions of new data based on Naïve Bayes Fit algorithm.

`Answer = predict(mdl, Xpredict)`

Returns the predicted label which corresponds to Xpredict based on the training model mdl.

Appendix B: MATLAB Code

Section 1: Yale Faces

```
% Khrisna Kamarga
% AMATH 482 - Homework 4
clear all; close all; clc;

cd 'C:\Users\Khrisna Adi Kamarga\Desktop\AMATH 482\HW4'
cd 'yalefaces';

files = dir('*.*');
files(1:2) = [];
for i=1:length(files)
    eval(['yalefaces' num2str(i) ' = imread(files(i).name);']);
end

cd ..
cd 'CroppedYale'

files = dir('*.*');
files(1:2) = [];
for i=1:9
    eval(['cd yaleB0' num2str(i)]);
    subFiles = dir('*.*');
    subFiles(1:2) = [];
    for j=1:length(subFiles)
        eval(['yaleB' num2str(i) '_' num2str(j) ' = imread(subFiles(j).name);']);
    end
    cd ..
end
for i=[10:13, 15:length(files)+1]
    eval(['cd yaleB' num2str(i)]);
    subFiles = dir('*.*');
    subFiles(1:2) = [];
    for j=1:length(subFiles)
        eval(['yaleB' num2str(i) '_' num2str(j) ' = imread(subFiles(j).name);']);
    end
    cd ..
end
cd ..
save loaded

%% Cropped
clear all; close all; clc;
load loaded

X = [];
for i=[1:13, 15:length(files)+1]
    for j=1:length(subFiles)
        eval(['currImage = reshape(yaleB' num2str(i) '_' num2str(j) ', [1 192*168]);']);
        X = [X; currImage];
    end
end
clearvars -except X
%% 1)
clc;
X = double(X);
[m,n]=size(X); % compute data size
[U S V] = svd(X/sqrt(n-1), 'econ');
lambda=diag(S); % produce diagonal variances
Y=U.'*X; % produce the principal components projection
%% 2)
close all; clc;
```

```
figure(1)
plot(1:length(lambda), lambda,'rx');
title("Energy Plot")
xlabel("Principal Component")
ylabel("Energy")

figure(3)
for i=1:12
    subplot(3,4,i)
    pcolor(flipud(reshape(Y(i,:), 192, 168))), shading interp, colormap hot
end

%% 3)
clc;

% Reducing the rank
r = 80;
Unew = U(:,1:r); Snew = S(1:r,1:r); Vnew = V(:,1:r);
Xr = Unew*Snew*Vnew';

figure(4)
for i=1:12
    subplot(3,4,i)
    pcolor(flipud(reshape(Xr(i+200*(i-1),:), 192, 168))), shading interp, colormap hot
end

figure(2)
for i=1:12
    subplot(3,4,i)
    pcolor(flipud(reshape(X(i+200*(i-1),:), 192, 168))), shading interp, colormap hot
end

%% 4) - Repeat 1) 2) 3) for uncropped

clear all; close all; clc;
load loaded

X = [];
for i=1:length(files)
    eval(['currImage = reshape(yalefaces' num2str(i) ', [1 243*320]);']);
    X = [X; currImage];
end
clearvars -except X
%% 1)
clc;
X=double(X);
[m,n]=size(X); % compute data siz
[U S V] = svd(X/sqrt(n-1), 'econ');
lambda=diag(S); % produce diagonal variances
Y=U.'*X; % produce the principal components projection
%% 2)
close all; clc;
figure(1)
plot(1:length(lambda), lambda,'rx');
title("Energy Plot")
xlabel("Principal Component")
ylabel("Energy")


figure(3),
for i=1:12
    subplot(3,4,i)
    pcolor(flipud(reshape(Y(i,:), 243, 320))), shading interp, colormap hot
end

%% 3)
```

```
clc;

% Reducing the rank
r = 32;
Unew = U(:,1:r); Snew = S(1:r,1:r); Vnew = V(:,1:r);
Xr = Unew*Snew*Vnew';

figure(4)
for i=1:12
    subplot(3,4,i)
    pcolor(flipud(reshape(Xr(i+2*(i-1),:), 243, 320))), shading interp, colormap hot
end

figure(2)
for i=1:12
    subplot(3,4,i)
    pcolor(flipud(reshape(X(i+2*(i-1),:), 243, 320))), shading interp, colormap hot
end
```

Section 2: Music Classification

```
%% Music Classification
clear all; close all; clc;
load automate
eval(['display("running Test' num2str(folder) '");']);
eval(['cd test' num2str(folder)])
files = dir('*.*');
files(1:2) = [];
for i=1:5
    eval(['test1_1_' num2str(i) ' = audioread(files(i).name);']);
end
for i=6:10
    eval(['test1_2_' num2str(i-5) ' = audioread(files(i).name);']);
end
for i=11:15
    eval(['test1_3_' num2str(i-10) ' = audioread(files(i).name);']);
end
cd ..
%% X Preparation by Reading Clips
display("preparing big X");

Fs = 44000; % sampling frequency
downSample = 8; % down sampling ratio
Fs = Fs/downSample; % new sampling frequency
T = 1/Fs; % sampling rate
fiveSeconds = 5/T; % data points for 5 seconds
X = [];
labelLength = [];
len = 0;
for i=1:3
    for j=1:5
        eval(['to = 1:length(test1_' num2str(i) '_' num2str(j) ');']);
        t = linspace(min(to),max(to), length(to)/downSample);
        samples = 20;
        eval(['temp = interp1(to, test1_' num2str(i) '_' num2str(j) ', t, ''linear'');']);
        for k=1:samples
            start = 5*fiveSeconds + round(k*fiveSeconds); % the start of sampling 5 seconds every
40 seconds
```

```
            eval(['clip' num2str(i) '_' num2str(j) '_' num2str(k) ' = temp(start:start+fiveSeconds-
1);']);
            eval(['clip' num2str(i) '_' num2str(j) '_' num2str(k) ' = mean(clip' num2str(i) '_'
num2str(j) '_' num2str(k) ',1);']);
            eval(['X = [X; clip' num2str(i) '_' num2str(j) '_' num2str(k) '];']);
        end
        len = len + samples; %amount of data points for each group
    end
    labelLength = [labelLength len];
end

%% Spectrogram Generation
clearvars -except X Fs labelLength;
display("Preparing Spectrograms");

allGabor = [];
for i=1:size(X,1)
    v = X(i,:); % current clip
    t = (1:length(v))/Fs; % making time vector (length Fs)
    L = max(t); n = length(t); % prepare the variables for fft
    k=(2*pi/L)*[0:n/2-1 -n/2:-1];

    t_sample = 0.1; %sampling rate

    tslide = 0:t_sample:L; % sampling time
    twindow = 20; % the width of the super gaussian
    spc=[]; %matrix of all the wavelets
    for j=1:length(tslide)
        g = exp(-(twindow*(t-tslide(j))).^10); % super gaussian
        vf=g.*v;
        yft=fft(vf);
        spc=[spc;abs(fftshift(yft))];
    end
    spc = spc.';
    spc(1:round(size(spc, 1)/2),:) = [];
%     %visualizing the spectrogram
%     pcolor(spc), shading interp, colormap(hot)
%     drawnow
    [m n] = size(spc);
    allGabor = [allGabor; reshape(spc, 1, m*n)];
    if (mod(i,10) == 0 | i == size(X,1))
        fprintf('%f percent completed \n', i/size(X,1)*100);
    end
end

%% SVD
clearvars -except allGabor labelLength;
display("Performing SVD")

n = size(allGabor, 1);
[U S V] = svd(allGabor, 'econ');
lambdaBig=diag(S).^2; % produce diagonal variances
Y=U.'*allGabor; % produce the principal components projection
% allGabor=allGabor-mean(allGabor(:));
% figure(1)
% plot(1:length(lambdaBig), lambdaBig,'rx');
% title("Energy Plot")
% xlabel("Principal Component")
% ylabel("Energy")

%% Training Data Preparation
display("Preparing Training Data");

Xtrain = [];
for i=1:4
    projection = [];
```

```
    for j=1:size(allGabor, 1)
        projection = [projection; dot(allGabor(j,:),Y(i,:))];
    end
    Xtrain = [Xtrain projection];
end

% figure(2)
% hold on
% plot(Xtrain(1:labelLength(1),1), Xtrain(1:labelLength(1), 2), 'rx');
% plot(Xtrain(labelLength(1)+1:labelLength(2),1), Xtrain(labelLength(1)+1:labelLength(2), 2),
'bo');
% plot(Xtrain(labelLength(2)+1:end,1), Xtrain(labelLength(2)+1:end, 2), 'k.');
% legend group1 group2 group3

%% Naive Bayes Training Model
display("Creating Naive Bayes Training Model");

label = [];
for i=1:labelLength(1)
    label = [label; "group1"];
end
for i=labelLength(1)+1:labelLength(2)
    label = [label; "group2"];
end
for i=labelLength(2)+1:length(Xtrain)
    label = [label; "group3"];
end

mdl = fitcnb(Xtrain, label);

%% Resample
display("Resampling new clips")

load automate

eval(['cd test' num2str(folder)])

files = dir('*.*');
files(1:2) = [];
for i=1:5
    eval(['test1_1_' num2str(i) ' = audioread(files(i).name);']);
end
for i=6:10
    eval(['test1_2_' num2str(i-5) ' = audioread(files(i).name);']);
end
for i=11:15
    eval(['test1_3_' num2str(i-10) ' = audioread(files(i).name);']);
end
cd ..

Fs = 44000; % sampling frequency
downSample = 8; % down sampling ratio
Fs = Fs/downSample; % new sampling frequency
T = 1/Fs; % sampling rate
fiveSeconds = 5/T; % data points for 5 seconds
X = [];
labelLength = [];
len = 0;
for i=1:3
    for j=1:5
        eval(['to = 1:length(test1_' num2str(i) '_' num2str(j) ');']);
        t = linspace(min(to),max(to), length(to)/downSample);
        samples = 3;
        eval(['temp = interp1(to, test1_' num2str(i) '_' num2str(j) ', t, ''linear'');']);
        for k=1:samples
```

```
            start = 20*fiveSeconds + round(k*fiveSeconds*3); % the start of sampling 5 seconds
every 40 seconds
            eval(['clip' num2str(i) '_' num2str(j) '_' num2str(k) ' = temp(start:start+fiveSeconds-
1);']);
            eval(['clip' num2str(i) '_' num2str(j) '_' num2str(k) ' = mean(clip' num2str(i) '_'
num2str(j) '_' num2str(k) ',1);']);
            eval(['X = [X; clip' num2str(i) '_' num2str(j) '_' num2str(k) '];']);
        end
        len = len + samples; %amount of data points for each group
    end
    labelLength = [labelLength len];
end

%% Regenerate Spectrogram
clearvars -except X Fs labelLength mdl Xtrain lambdaBig;
display("Regenerating Spectrograms")

allGabor = [];
for i=1:size(X,1)
    v = X(i,:); % current clip
    t = (1:length(v))/Fs; % making time vector (length Fs)
    L = max(t); n = length(t); % prepare the variables for fft
    k=(2*pi/L)*[0:n/2-1 -n/2:-1];

    t_sample = 0.1; %sampling rate

    tslide = 0:t_sample:L; % sampling time
    twindow = 20; % the width of the super gaussian
    spc=[]; %matrix of all the wavelets
    for j=1:length(tslide)
        g = exp(-(twindow*(t-tslide(j))).^10); % super gaussian
        vf=g.*v;
        yft=fft(vf);
        spc=[spc;abs(fftshift(yft))];
    end
%     visualize the spectrogram
%     pcolor(tslide,fftshift(k)/(2*pi),spc.'), shading interp, colormap(hot)
    spc = spc.';
    spc(1:round(size(spc, 1)/2),:) = [];
    [m n] = size(spc);
    allGabor = [allGabor; reshape(spc, 1, m*n)];
    if (mod(i,10) == 0 | i == size(X,1))
        fprintf('%f percent completed \n', i/size(X,1)*100);
    end
end

%% SVD of the Resample
clearvars -except allGabor labelLength label mdl Xtrain lambdaBig;
display("SVDing the resampled data")

n = size(allGabor, 1);
[U S V] = svd(allGabor, 'econ');
lambda=diag(S).^2; % produce diagonal variances
Y=U.'*allGabor; % produce the principal components projection
% allGabor=allGabor-mean(allGabor(:));
% figure(3)
% plot(1:length(lambda), lambda,'rx');
% title("Energy Plot")
% xlabel("Principal Component")
% ylabel("Energy")

clc;
Xpredict = [];
for i=1:4
    projection = [];
    for j=1:size(allGabor, 1)
```

```
        projection = [projection; dot(allGabor(j,:),Y(i,:)')];
    end
    Xpredict = [Xpredict projection];
end
%%
% figure(2)
% hold on
% plot(Xpredict(1:labelLength(1),1), Xpredict(1:labelLength(1), 2), 'rx', 'LineWidth', 5);
% plot(Xpredict(labelLength(1)+1:labelLength(2),1), Xpredict(labelLength(1)+1:labelLength(2), 2),
'bo', 'LineWidth', 5);
% plot(Xpredict(labelLength(2)+1:end,1), Xpredict(labelLength(2)+1:end, 2), 'k^', 'LineWidth', 5);
% legend group1 group2 group3

%%
display("Predicting")

answer = [];
correctAnswer = 0;
for i=1:length(Xpredict)
    currAnswer = predict(mdl, Xpredict(i,:));
    currAnswer = string(currAnswer);
    answer = [answer currAnswer];
    if (i <= 15 & currAnswer == 'group1' | ...
        i> 15 & i <= 30 & currAnswer == 'group2' | ...
        i> 30 & currAnswer == 'group3')
        correctAnswer = correctAnswer +1;
    end
end

accuracy = correctAnswer/length(Xpredict) * 100;
display(answer)
```

Supporting Script: Automating Data Collection for Music Classification

```
% Khrisna Kamarga
% AMATH482 - HW4 Automate
close all; clear all; clc;
display("Test1")
folder = 1;
save automate
Kamarga_AMATH482_HW4
%%
save log_test1

close all; clear all; clc;
display("Test2")
folder = 2;
save automate
Kamarga_AMATH482_HW4
save log_test2
%%
close all; clear all; clc;
display("Test3")
folder = 3;
save automate
Kamarga_AMATH482_HW4
save log_test3
close all; clear all; clc;

%%
clear all; close all; clc
load log_test1

figure(1)
plot(1:length(lambdaBig), lambdaBig,'rx');
```

```
title("Energy Plot")
xlabel("Principal Component")
ylabel("Energy")
grid on

figure(2)
plot3(Xtrain(1:100,1), Xtrain(1:100, 2), Xtrain(1:100, 3), 'rx');
hold on
plot3(Xtrain(101:200,1), Xtrain(101:200, 2), Xtrain(101:200, 3), 'bo');
hold on
plot3(Xtrain(201:end,1), Xtrain(201:end, 2), Xtrain(201:end, 3), 'k.');
legend group1 group2 group3
hold on
plot3(Xpredict(1:labelLength(1),1), Xpredict(1:labelLength(1), 2), Xpredict(1:labelLength(1), 3),
'rx', 'LineWidth', 5);
hold on
plot3(Xpredict(labelLength(1)+1:labelLength(2),1), Xpredict(labelLength(1)+1:labelLength(2), 2),
Xpredict(labelLength(1)+1:labelLength(2), 3), 'bo', 'LineWidth', 5);
hold on
plot3(Xpredict(labelLength(2)+1:end,1), Xpredict(labelLength(2)+1:end, 2),
Xpredict(labelLength(2)+1:end, 3), 'k^', 'LineWidth', 5);
legend group1 group2 group3
grid on
```