HW4: Music Classification

AMATH 482 | Celeste Becker

Abstract

The objective of this homework is to explore the basics of machine learning by writing code that can classify a song by what genre it is using a five second sample clip. To program the classifier, we will use spectrograms, Principal Component Analysis, (PCA) and Linear Discriminant Analysis (LDA).

Sec. I. Introduction and Overview

In this problem we need to collect music data. Training data and test data. This is collected by taking five second clips from songs of various genres and artists on Spotify. These are turned into .mp3 files that are fed into the MATLAB program. We create a spectrogram of each song and send the spectrogram data into the PCA algorithm. The goal is to build a statistical testing algorithm that can accurately identify a new five second clip of music and sort it into the correct genre (or classify as it as by correct artist).

This homework has three different tests:

Test 1: Band (artist) Classification. Here we will take music from three different artists, each from a unique genre. Duke Ellington for jazz, J. Cole for hip hop, and Metallica for metal. The test data will be songs from each artist.

Test 2: Same genre classification. Here we take music from three different artists all from the same genre of hip hop. We are using data from J. Cole, Chance the Rapper, and Kanye West. The test data will also be different songs from each artist.

Test 3: Genre Classification. For this test, we are using data from many different artists in the same genre to build a classifier. The genres chosen are Jazz, Hip Hop, and Metal. The test data will be songs in each genre from a variety of artists.

For each test, we will send in test data and report the accuracy of the program to correctly classify the new songs.

To do this, we will take spectrogram data from several classes of music data. Then we will use SVD and PCA to and project the data onto the Principal components, and use LDA to maximize the between class variance, and minimize the inter-class variance. This way we can find threshold values between the different classes of music. Then we will run test data through our classifier and find the percent accuracy of each test.

Sec. II. Theoretical Background

The main mathematical method focused on in this program is *LDA, Linear Discriminant Analysis.* The information below is referenced from chapter 17 of *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data* by J. Nathan Kutz.

To implement the classifier algorithm, we will use SVD composition to extract the key components: U, Σ, and V.

We will set a variable called feature, that determines how many principal components will be considered. We then limit the number of columns of U and rows of Σ *V that are extracted by this feature value.

LDA – Linear Discrimination Analysis: The goal of LDA is to find a projection that maximizes the distance between inter-class data and minimizes the distance between the intra-class data.

The equation for the between class scatter matrix: $S_B = \sum_{n=1}^{n}(\mu_n - \mu)(\mu_n - \mu)^T$ (Equation 1)

The equation for the within class scatter matrix: $S_W = \sum_{j=1}^{n}\sum_{X}(X - \mu)(X - \mu)^T$ (Equation 2)

For the above two equations, μ is the mean across all classes, and μ_n is the mean of a specific class.

To find the vector W that we want to project the data onto, we solve the generalized eigenvalue problem:

$S_B w = \lambda S_W w$ (Equation 3)

In the above equation, the maximum eigenvalue λ and its associated eigenvector w gives the quantity of interest and the projection basis.

Sec. III. Algorithm Implementation and Development

Each section labeled here corresponds to the commented section with the same name in *Appendix B. Matlab Codes* at the end of this report. Code reference lines are also included. This program has been split up into four files. One for each test. Each mention of TEST is a different file.

Each TEST program follows the same logic, just for different data sets.

Spectrogram function: The spectrogram function takes in a music file, and outputs a column vector with the spectrogram data for that song. The sampling rate Fs is lowered by a factor of 10 in order to make the program load faster. This means we are taking every tenth data point. See: TEST 1 – DEFINING FUNCTIONS -lines (191-234)

Trainer function: This function takes in spectrogram data for three different data sets, and the number of features. This function performs the SVD and does the PCA projection of each class of data onto principal components. It is also what does the Linear Discriminant Analysis and

finds the thresholds. This function outputs the U, S, and V from the SVD, and the first and second thresholds (th1 and th2). It also returns the w vector, and the sorted projected class data. See: **TEST 2 – DEFINING FUNCTIONS - lines (236 - 311)**

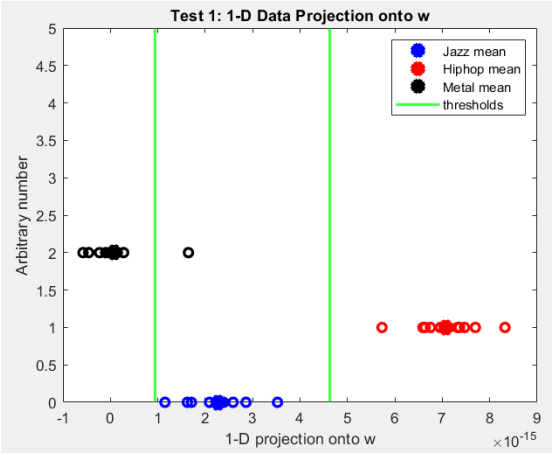
First, the program loads in the different music data, and uses the Spectrogram function to create spectrograms of each song. Next, we set the features variable, which represents the number of principal components we want to use and load the spectrogram data into the Trainer function. See: **TEST 1 – Training Data – lines (61 – 74)**

By plotting the sorted one-dimensional projection of the different classes of music data and trying out different values for the feature variable, we can decide what value of feature best sorts the data. See: **TEST 1 – Plot Projections onto w and mean(*) – lines (158-188)**

Then, for each test dataset, we create a spectrogram of the test song, and multiply it by the U matrix from the Training dataset. We then project this testSong variable onto the vector w from the Training dataset, and set it equal to a variable pval. This pval is what classifies the test song. Depending on this value, and it's relationship to the thresholds, we can see where the program classified the new song. The test data is organized by class within the folder, so there are multiple counter variables to keep track of the accuracy of the program. See: **TEST 1 – Test Data – lines (75-149)**

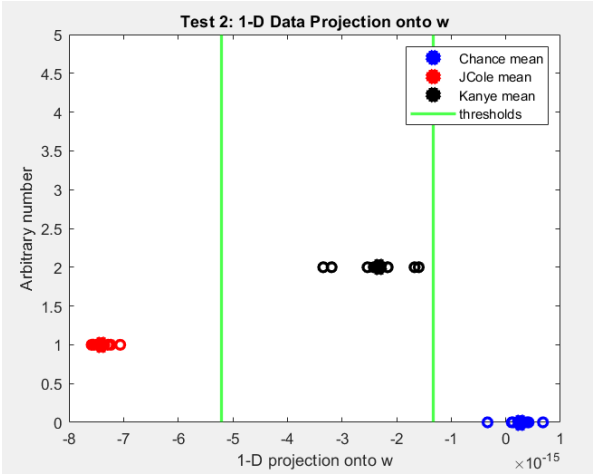
Then, we print out to the console the percent accuracy of the test data. See: **TEST 1- Print out Percent Accuracy – lines (150-157).**

Figure 1 – Test 1



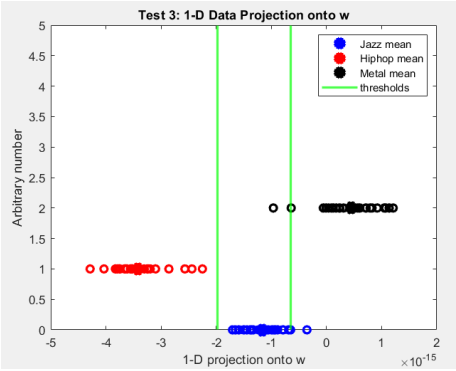

```
Percent Accuracy for test Metal songs = 90
Percent Accuracy for test Jazz songs = 60
Percent Accuracy for test Hip Hop songs = 60
Total Percent Accuracy = 70
```

Figure 2 – Test 2




```
Percent Accuracy for test JCole songs = 90
Percent Accuracy for test Kanye West songs = 40
Percent Accuracy for test Chance songs = 20
Total Percent Accuracy = 50
```

Figure 3 – Test 3




```
Percent Accuracy for test Metal songs = 20
Percent Accuracy for test Jazz songs = 50
Percent Accuracy for test Hip Hop songs = 80
Total Percent Accuracy = 50
```

Sec. IV. Computational Results

Test 1: See *Figure 1 – Test 1* above. For this test we had training music data from three different artists, within three different genres. Duke Ellington for jazz, J. Cole for hip hop, and Metallica for metal. For the training data, is 10 clips from each genre, so 30 clips for all the training data. The test data is 10 songs from each artist. Setting features to 24, you can see the clear separation between genres in the graph. The final total percent accuracy across all genres was 70%. This is a pretty good number for how little training data is used.

Test 2: See *Figure 2 – Test 2* above . This is for the Same genre classification. Here we take music from three different artists all from the same genre of hip hop. We are using data from J. Cole, Chance the Rapper, and Kanye West. Each with 10 music clips per artist, so 30 training clips in total. The test data is 10 different songs from each artist, so 30 test clips in total. We set the number of features equal to 25. The final total percent accuracy across all genres was 50%. This is much worse than the first test. One reason is probably because there just isn't enough training data. It's also harder to distinguish between classes because since they are all from the same genre the clips sound very similar. Having a much larger training data set would probably improve the overall percent accuracy.

Test 3: See *Figure 3 – Test 3* above . For this test, we are using data from many different artists in the same genre to build a classifier. The genres chosen are Jazz, Hip Hop, and Metal. The test data will be songs in each genre from a variety of artists. For each genre there were 30 training clips. The test data is 10 different songs from each genre, so 30 test clips in total of test data. We set the number of features equal to 74. We need to have a larger number of features because since the training data is not from a single artist, but from many different bands, each clip can have a big variance in sound, even if they are in the same genre. This makes it much harder to classify the new test data. The final total percent accuracy across all genres was 50%. This is much worse than the first test, and equal to the second test. One reason is probably

because there just isn't enough training and testing data. With a few orders of magnitude more for training and testing data, we should be able to get results closer to Test 1.

Sec. V. Summary and Conclusions

In summary, we were able to use LDA to create a classifier program to classify a new clip of music by genera or by artist. Using multiple orders of magnitude more training data in the future would let us project onto fewer principal components, and get better percent accuracy results.

Appendix A. MATLAB functions used and brief implementation explanation

svd

- [U,S,V] = svd(A)**.**
- Computes the SVD algorithm in matlab
- *Implementation*: Used to convert the Gabor transformed signal from the time domain into the frequency domain in the spectrogram function. Ex: TEST 1 – line 248: [U,S,V] = svd(allMusic,'econ');

audioread

- [Y,FS] = audioread(filename)
- reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.
- *Implementation*: Used to read in all music data. Ex: TEST 1 – line 197: [y,Fs] = audioread(songPath);

eig

- [V,D] = eig(A,B)
- returns diagonal matrix D of generalized eigenvalues and full matrix V whose columns are the corresponding right eigenvectors, so that A*V = B*V*D
- *Implementation*: Used to solve (Equation 3). Ex: Test 1 – line 282: [V2,D] = eig(Sb,Sw);

Sort

- B = sort(A,dim)
- returns the sorted elements of A along dimension dim
- *Implementation*: Used to sort the data projected onto w. Ex: Test 1 – line 292: sortJazz = sort(vjazz)

Appendix B. MATLAB codes

Each test is contained in its own file.

TEST 1

```
%% TEST 1
close all; clear all; clc;

%% Get Spectrogram Data - lines (4-60)

%Jazz Data - Duke Ellington
songFiles = dir('C:\Users\celes\Documents\UW\AMATH 482\HW\HW
4\DataTest1\DukeEllington\*.mp3');
jazzSpecData = [];

%loop through song files in folder
for k = 1:length(songFiles)

    %this gets the file path of the song
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);

    %prints song name
    fprintf('Song: #%d = %s\n', k, songFiles(k).name);

    %stores spectrogram data for each song in a column
    jazzSpecData(:,k) = spectogramCol(songPath);

end


%Hip Hop Data - J. Cole
songFiles = dir('C:\Users\celes\Documents\UW\AMATH 482\HW\HW
4\DataTest1\JCole\*.mp3');
hiphopSpecData = [];

%loop through song files in folder
for k = 1:length(songFiles)

    %this gets the file path of the song
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);

    %prints song name
    fprintf('Song: #%d = %s\n', k, songFiles(k).name);

    %stores spectrogram data for each song in a column
```

```
    hiphopSpecData(:,k) = spectogramCol(songPath);

end


%Metal Data - Metallica
songFiles = dir('C:\Users\celes\Documents\UW\AMATH 482\HW\HW
4\DataTest1\Metalica\*.mp3');
metalSpecData = [];
%loop through song files in folder
for k = 1:length(songFiles)

    %this gets the file path of the song
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);

    %prints song name
    fprintf('Song: #%d = %s\n', k, songFiles(k).name);

    %stores spectrogram data for each song in a column
    metalSpecData(:,k) = spectogramCol(songPath);

end

%% Training Data - lines (61-74)
%set the feature number equal to the number of principal
components
%that will be considered
feature = 24;

%Call the trainer function, which does the SVD, and extracts
principal
%components, and returns the threshold and the projection of the
different
%classes of data onto the w vector
[U,S,V,th1,th2,w,sortJazz,sortHiphop,sortMetal] =
trainer(jazzSpecData, hiphopSpecData, metalSpecData, feature);

meanJazz = mean(sortJazz,2);
meanHiphop = mean(sortHiphop,2);
meanMetal = mean(sortMetal,2);

%% Test Data lines(75-149)
%The training data is organized in a folder. Below we go through
the
%traing data for each genre and count how many times the program
correctly
```

```
%classified the song, and then returns the
songFiles = dir('C:\Users\celes\Documents\UW\AMATH 482\HW\HW
4\DataTest1\TestData\*.mp3');

%Metal Test Data
songData = [];
counterMetal = 0;
%Loop Through All Songs in Folder
for k = 1:10
    %this gets the file path of the song
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);

    %prints song name
    fprintf('Test Song: #%d = %s\n', k, songFiles(k).name);
    songData= spectogramCol(songPath);

    testSong = U'*songData;
    pval = w'*testSong;
    testClassfications(k) = pval;

    if (pval > th1)
        counterMetal = counterMetal+1;
    end
end
 percentCorrectMetal = 100*(counterMetal/10);


%Jazz Test Data
songData = [];
counterJazz = 0;
for k = 11:20
    %this gets the file path of the song
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);

    %prints song name
    fprintf('Test Song: #%d = %s\n', k, songFiles(k).name);
    songData= spectogramCol(songPath);

    testSong = U'*songData;
    pval = w'*testSong;
    testClassfications(k) = pval;

    if ((pval > th1) && (pval < th2))
        counterJazz = counterJazz+1;
    end
```

```
end
 percentCorrectJazz = 100*(counterJazz/10);


%Hip Hop Test Data
songData = [];
counterHiphop = 0;
%only  one thing in this for loop so get it to work for one
for k = 21:30
    %this gets the file path of the song
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);

    %prints song name
    fprintf('Test Song: #%d = %s\n', k, songFiles(k).name);
    songData= spectogramCol(songPath);

    testSong = U'*songData;
    pval = w'*testSong;
    testClassfications(k) = pval;

    if (pval > th2)
        counterHiphop = counterHiphop+1;
    end
end

 percentCorrectHiphop = 100*(counterHiphop/10);
 totalPercentCorrect =
100*((counterJazz+counterMetal+counterHiphop)/length(songFiles))
;

 %% Print out Percent Accuracy - lines (150-157)
fprintf('\n');
fprintf('\n');
fprintf('Percent Accuracy for test Metal songs = %d \n',
percentCorrectMetal);
fprintf('Percent Accuracy for test Jazz songs = %d \n',
percentCorrectJazz);
fprintf('Percent Accuracy for test Hip Hop songs = %d \n',
percentCorrectHiphop);
fprintf('Total Percent Accuracy = %d \n', totalPercentCorrect);

%% Plot - projections onto w and mean(*)- lines(158 - 188)
figure(1)
%jazz = blue
plot(sortJazz,zeros(),'ob','Linewidth',2);
hold on
```

```
m1 = plot(meanJazz,0,'*b','Linewidth',10,'DisplayName','jazz
mean');

%hiphop = red
hold on
plot(sortHiphop,1,'or','Linewidth',2);
hold on
m2 =
plot(meanHiphop,1,'*r','Linewidth',10,'DisplayName','hiphop');

%metal = black
hold on
plot(sortMetal,ones()*2,'ok','Linewidth',2);
hold on
m3 =
plot(meanMetal,2,'*k','Linewidth',10,'DisplayName','metal');

%plot thresholds
hold on
m4 = xline(th1,'g','Linewidth',2);
hold on
xline(th2,'g','Linewidth',2);

ylim([0 5])
title('Test 1: 1-D Data Projection onto w');
ylabel('Arbitrary number');
xlabel('1-D projection onto w');
legend([m1,m2,m3,m4],'Jazz mean','Hiphop mean','Metal
mean','thresholds')

%% -------- DEFINING FUNCTIONS ------ lines ( 189-314)

%Spectogram function - lines (191-234)
%Takes in data and computs the spectrogram, returns
%the data reshaped into one column vector.
function data = spectogramCol(songPath)

        %geting music / spectogram data
        [y,Fs] = audioread(songPath);

         %Lower sampling rate by a factor of 10
         Fs = Fs/10;
         song = y(1:Fs:5,:);
         S = song'; %Signal S equal to the song data amplitude
        L=5;
        time=length(song)/Fs; % total recorded time in seconds
        %time=length(L)/Fs; % total recorded time in seconds
```

```
        %t = (1:L)/Fs; % vector of time in seconds

        n=Fs*L; % n is the number of data points in signal
        t = linspace(0,L,n);


        k=(1/L)*[0:n/2-1 -n/2:-1]; %EVEN # of frequencys in Hz
        ks=fftshift(k);

        %Take the Spectogram of the song
        tslide=0:0.1:L; %Descrete time vector
        Sgt_spec = zeros(length(tslide),n);

        %gabor filter width
        a = 150;

        for jj=1:length(tslide)
            %applying gabor filter
            g=exp(-a*(t-tslide(jj)).^2);
            Sg=g.*S;
            Sgt=fft(Sg);%fourier transforming to the freuqency
domain

            %storing all the filtered signal at that time in the
spectogram vector
            Sgt_spec(jj,:) = fftshift(abs(Sgt));

        end

        data = reshape(Sgt_spec,
(length(tslide)*length(Sgt_spec)), 1);
end



%Trainer fucntion
%is used on the training data and performs the SVD and LDA
%also computs the first threshold (th1) and second threshold
(th2)
function [U,S,V,th1,th2,w,sortJazz,sortHiphop,sortMetal] =
trainer(jazz0,hiphop0,metal0,feature)


    nj = size(jazz0,2); nh = size(hiphop0,2); nm =
size(metal0,2);
    allMusic = [jazz0 hiphop0 metal0];
```

```
%SVD
[U,S,V] = svd(allMusic,'econ');

% PCA projection onto principal components
musicProjection = S*V';
U = U(:,1:feature);
Jazz = musicProjection(1:feature,1:nj);
Hiphop = musicProjection(1:feature,nj+1:nj+nh);
Metal = musicProjection(1:feature,nh+nm+1:nh+nm+nj);

mj = mean(Jazz,2);
mh = mean(Hiphop,2);
mm = mean(Metal,2);
mTotal = mean(allMusic(1:feature),2);

% LDA -- WITHIN CLASS VARIANCES
Sw = 0;
for k=1:nj
    Sw = Sw + (Jazz(:,k)-mj)*(Jazz(:,k)-mj)';
end
for k=1:nh
    Sw = Sw + (Hiphop(:,k)-mh)*(Hiphop(:,k)-mh)';
end
for k=1:nm
    Sw = Sw + (Metal(:,k)-mm)*(Metal(:,k)-mm)';
end

% BETWEEN CLASS VARIANCES
SbJazz = (mj-mTotal)*(mj-mTotal)';
SbHiphop = (mh - mTotal)*(mh - mTotal)';
SbMetal = (mm - mTotal)*(mh - mTotal)';
Sb = SbJazz + SbHiphop + SbMetal;


%linear discriminant analysis
[V2,D] = eig(Sb,Sw);
[~,ind] = max(abs(diag(D)));
w = V2(:,ind); w = w/norm(w,2);

vjazz = w'*Jazz;
vhiphop = w'*Hiphop;
vmetal = w'*Metal;

sortJazz = sort(vjazz);
sortHiphop = sort(vhiphop);
sortMetal = sort(vmetal);
```

```
    %THRESHOLDS
    %finding threshold 1 - between metal and jazz
    t1 = length(sortMetal);
    t2 = 1;
    while sortMetal(t1)>sortJazz(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    %threshold 1
    th1 = (sortMetal(t1)+sortJazz(t2))/2;

    %finding threshold 2 - between hiphop and metal
    t1 = length(sortJazz);
    t2 = 1;
    while sortJazz(t1)>sortHiphop(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    %threshold 2
    th2 = (sortJazz(t1)+sortHiphop(t2))/2;

end
```

TEST 2

```
%% TEST 2
close all; clear all; clc;

%% Get Spectrogram Data

%Chance the Rapper Spectogram data
songFiles = dir('C:\Users\celes\Documents\UW\AMATH 482\HW\HW
4\DataTest2\Chance\*.mp3');
chanceSpecData = [];

for k = 1:length(songFiles)

    %this gets the file path of the song
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);

    %prints song name
    fprintf('Song: #%d = %s\n', k, songFiles(k).name);
    data = spectogramCol(songPath);
    chanceSpecData(:,k) = data;

end

%Jcole spectogram data
songFiles = dir('C:\Users\celes\Documents\UW\AMATH 482\HW\HW
4\DataTest2\JCole\*.mp3');
%Loop Through All Songs in Folder
jcoleSpecData = [];

for k = 1:length(songFiles)
```

```
    %this gets the file path of the song
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);

    %PRINTS song name
    fprintf('Song: #%d = %s\n', k, songFiles(k).name);
    data = spectogramCol(songPath);
    jcoleSpecData(:,k) = data;

end

% Kanye west spectogram data
songFiles = dir('C:\Users\celes\Documents\UW\AMATH 482\HW\HW
4\DataTest2\Kanye\*.mp3');
%Loop Through All Songs in Folder
kanyeSpecData = [];

for k = 1:length(songFiles)

    %this gets the file path of the song
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);

    %PRINTS song name
    fprintf('Song: #%d = %s\n', k, songFiles(k).name);
    data = spectogramCol(songPath);
    kanyeSpecData(:,k) = data;

end

%% Training Data
%set the feature number equal to the number of principal
components
%that will be considered
feature = 25;

%Call the trainer function, which does the SVD, and extracts
principal
%components, and returns the threshold and the projection of the
different
%classes of data onto the w vector
[U,S,V,th1,th2,w,sortChance,sortJCole,sortKanye,musicProjection]
= trainer(chanceSpecData, jcoleSpecData, kanyeSpecData,
feature);
size(musicProjection)
```

```
meanChance = mean(sortChance,2);
meanJCole = mean(sortJCole,2);
meanKanye = mean(sortKanye,2);

%% Test Data - lines (71
%The training data is organized in a folder. Below we go through
the
%traing data for each genre and count how many times the program
correctly
%classified the song, and then returns the
songFiles = dir('C:\Users\celes\Documents\UW\AMATH 482\HW\HW
4\DataTest2\TestData\*.mp3');

%JCole test data
songData = [];
counterJCole = 0;
for k = 21:30
    %this gets the file path of the song
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);

    %prints song name
    fprintf('Test Song: #%d = %s\n', k, songFiles(k).name);
    songData= spectogramCol(songPath);

    testSong = U'*songData;
    pval = w'*testSong;
    testClassfications(k) = pval;

    %checks if the predicted value for the song is correct for
jcole
    if (pval > th1)
        counterJCole = counterJCole+1;
    end
end

 percentCorrectJCole = 100*(counterJCole/10);


%Kanye West test data
 songData = [];
counterKanye = 0;

for k = 11:20
    %this gets the file path of the song
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);
```

```
    %prints song name
    fprintf('Test Song: #%d = %s\n', k, songFiles(k).name);
    songData= spectogramCol(songPath);

    testSong = U'*songData;
    pval = w'*testSong;
    testClassfications(k) = pval;

    if ((pval > th1) && (pval < th2))
        counterKanye = counterKanye+1;
    end
end
 percentCorrectKanye = 100*(counterKanye/10);


%Chance the Rapper Test data
songData = [];
counterChance = 0;
%only  one thing in this for loop so get it to work for one
for k = 1:10
    %this gets the file path of the song
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);

    %prints song name
    fprintf('Test Song: #%d = %s\n', k, songFiles(k).name);
    songData= spectogramCol(songPath);

    testSong = U'*songData;
    pval = w'*testSong;
    testClassfications(k) = pval;

    if (pval > th2)
        counterChance = counterChance+1;
    end
end

 percentCorrectChance = 100*(counterChance/10);

 %calculates the total correct percentage
 totalPercentCorrect =
100*((counterKanye+counterJCole+counterChance)/length(songFiles)
);

%% Print out Percent Accuracy
fprintf('\n');
```

```
fprintf('\n');
fprintf('Percent Accuracy for test JCole songs = %d \n',
percentCorrectJCole);
fprintf('Percent Accuracy for test Kanye West songs = %d \n',
percentCorrectKanye);
fprintf('Percent Accuracy for test Chance songs = %d \n',
percentCorrectChance);
fprintf('Total Percent Accuracy = %d \n', totalPercentCorrect);

%% Plot - projections onto w and mean(*)
figure(1)
%chance = blue
plot(sortChance,zeros(),'ob','Linewidth',2);
hold on
m1 = plot(meanChance,0,'*b','Linewidth',10,'DisplayName','jazz
mean');

%jcole = red
hold on
plot(sortJCole,1,'or','Linewidth',2);
hold on
m2 =
plot(meanJCole,1,'*r','Linewidth',10,'DisplayName','hiphop');

%kanye = black
hold on
plot(sortKanye,ones()*2,'ok','Linewidth',2);
hold on
m3 =
plot(meanKanye,2,'*k','Linewidth',10,'DisplayName','metal');

%plot thresholds
hold on
m4 = xline(th1,'g','Linewidth',2);
hold on
xline(th2,'g','Linewidth',2);

ylim([0 5])
title('Test 2: 1-D Data Projection onto w');
ylabel('Arbitrary number');
xlabel('1-D projection onto w');
legend([m1,m2,m3,m4],'Chance mean','JCole mean','Kanye
mean','thresholds')

%% -------- DEFINING FUNCTIONS ------
```

```
%Spectogram function. Takes in data and computs the spectrogram,
returns
%the data reshaped into one column vector.
function data = spectogramCol(songPath)

        %geting music / spectogram data --> Change song files?
        [y,Fs] = audioread(songPath);

         %Lower sampling rate by a factor of 10
         Fs = Fs/10;
         song = y(1:Fs:5,:);
         S = song'; %Signal S equal to the song data amplitude
        L=5;
        time=length(song)/Fs; % total recorded time in seconds
        %time=length(L)/Fs; % total recorded time in seconds
        %t = (1:L)/Fs; % vector of time in seconds

        n=Fs*L; % n is the number of data points in signal
        t = linspace(0,L,n);


        k=(1/L)*[0:n/2-1 -n/2:-1]; %EVEN # of frequencys in Hz
        ks=fftshift(k);

        %Take the Spectogram of the song
        tslide=0:0.1:L; %Descrete time vector
        Sgt_spec = zeros(length(tslide),n);

        %gabor filter width
        a = 150;

        for jj=1:length(tslide)
            %applying gabor filter
            g=exp(-a*(t-tslide(jj)).^2);
            Sg=g.*S;
            Sgt=fft(Sg);%fourier transforming to the freuqency
domain

            %storing all the filtered signal at that time in the
spectogram vector
            Sgt_spec(jj,:) = fftshift(abs(Sgt));

        end

        data = reshape(Sgt_spec,
(length(tslide)*length(Sgt_spec)), 1);
end
```

```
%Trainer fucntion - lines (236-311)
%is used on the training data and performs the SVD and LDA
%also computs the first threshold (th1) and second threshold
(th2)
function [U,S,V,th1,th2,w,sortgenre1,sortgenre2,sortgenre3,
musicProjection] = trainer(genre01,genre02,genre03,feature)

    n1 = size(genre01,2); n2 = size(genre02,2); n3 =
size(genre03,2);
    allMusic = [genre01 genre02 genre03];

    %SVD
    [U,S,V] = svd(allMusic,'econ');

    % PCA projection onto principal components
    musicProjection = S*V';
    U = U(:,1:feature);
    Genre1 = musicProjection(1:feature,1:n1);
    Genre2 = musicProjection(1:feature,n1+1:n1+n2);
    Genre3 = musicProjection(1:feature,n2+n3+1:n2+n3+n1);

    m1 = mean(Genre1,2);
    m2 = mean(Genre2,2);
    m3 = mean(Genre3,2);
    mTotal = mean(allMusic(1:feature),2);

    % LDA -- WITHIN CLASS VARIANCES
    Sw = 0;
    for k=1:n1
        Sw = Sw + (Genre1(:,k)-m1)*(Genre1(:,k)-m1)';
    end
    for k=1:n2
        Sw = Sw + (Genre2(:,k)-m2)*(Genre2(:,k)-m2)';
    end
    for k=1:n3
        Sw = Sw + (Genre3(:,k)-m3)*(Genre3(:,k)-m3)';
    end

    % BETWEEN CLASS VARIANCES
    SbGenre1 = (m1-mTotal)*(m1-mTotal)';
    SbGenre2 = (m2 - mTotal)*(m2 - mTotal)';
    SbGenre3 = (m3 - mTotal)*(m2 - mTotal)';
    Sb = SbGenre1 + SbGenre2 + SbGenre3;

    % linear discriminant analysis
```

```
    [V2,D] = eig(Sb,Sw);
    [~,ind] = max(abs(diag(D)));
    w = V2(:,ind); w = w/norm(w,2);

    vgenre1 = w'*Genre1;
    vgenre2 = w'*Genre2;
    vgenre3 = w'*Genre3;

    sortgenre1 = sort(vgenre1);
    sortgenre2 = sort(vgenre2);
    sortgenre3 = sort(vgenre3);

    %THRESHOLDS
    %finding threshold 1 - between jazz and hiphop
    t1 = length(sortgenre2);
    t2 = 1;
    while sortgenre2(t1)>sortgenre3(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    %threshold 1
    th1 = (sortgenre2(t1)+sortgenre3(t2))/2;

    %finding threshold 2 - between hiphop and metal
    t1 = length(sortgenre3);
    t2 = 1;
    while sortgenre3(t1)>sortgenre1(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    %threshold 2
    th2 = (sortgenre1(t1)+sortgenre3(t2))/2;

end
```

TEST 3

```
%% TEST 3
close all; clear all; clc;

%% Get Spectrogram Data
songFiles = dir('C:\Users\celes\Documents\UW\AMATH 482\HW\HW
4\DataTest3\z Jazz Classics small\*.mp3');

%Jazz Data - various artists
jazzSpecData = [];
for k = 1:length(songFiles)
    %this gets the file path of the song
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);

    %PRINTS song name
    fprintf('JAZZ: #%d = %s\n', k, songFiles(k).name);
    data = spectogramCol(songPath);
    jazzSpecData(:,k) = data;
end

% Hip Hop Data - various artists
songFiles = dir('C:\Users\celes\Documents\UW\AMATH 482\HW\HW
4\DataTest3\z Hip Hop Drive small\*.mp3');
hiphopSpecData = [];

for k = 1:length(songFiles)
    %this gets the file path of the song
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);

    %PRINTS song name
    fprintf('HIPHOP: #%d = %s\n', k, songFiles(k).name);
    data = spectogramCol(songPath);
    hiphopSpecData(:,k) = data;

end


% Metal data - various artists
```

```
songFiles = dir('C:\Users\celes\Documents\UW\AMATH 482\HW\HW
4\DataTest3\z Old School Metal small\*.mp3');
metalSpecData = [];
for k = 1:length(songFiles)
    %this gets the file path of the song
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);

    %PRINTS song name
    fprintf('METAL: #%d = %s\n', k, songFiles(k).name);
    data = spectogramCol(songPath);
    metalSpecData(:,k) = data;

end

%% Training Data
%set the feature number equal to the number of principal
components
%that will be considered
feature = 74;

%Call the trainer function, which does the SVD, and extracts
principal
%components, and returns the threshold and the projection of the
different
%classes of data onto the w vector
[U,S,V,th1,th2,w,sortJazz,sortHiphop,sortMetal,musicProjection]
= trainer(jazzSpecData, hiphopSpecData, metalSpecData, feature);

meanJazz = mean(sortJazz,2);
meanHiphop = mean(sortHiphop,2);
meanMetal = mean(sortMetal,2);

%% Test Data
%The training data is organized in a folder. Below we go through
the
%traing data for each genre and count how many times the program
correctly
%classified the song, and then returns the
songFiles = dir('C:\Users\celes\Documents\UW\AMATH 482\HW\HW
4\DataTest3\TestData\*.mp3');

%Hip hop test data - 10 songs
songData = [];
counterHiphop = 0;
%only  one thing in this for loop so get it to work for one
for k = 21:30
```

```
    %this gets the file path of the song
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);

    %PRINTS song name
    fprintf('Test Song: #%d = %s\n', k, songFiles(k).name);
    songData= spectogramCol(songPath);

    testSong = U'*songData;
    pval1 = w'*testSong;
    testClassfications(k) = pval1;

    if (pval1 <= th1)
        counterHiphop = counterHiphop+1;
    end
end
 percentCorrectHiphop = 100*(counterHiphop/10);


%Jazz test data - 10 songs
songData = [];
counterJazz = 0;
%only  one thing in this for loop so get it to work for one
for k = 11:20
    %this gets the file path of the song
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);

    %PRINTS song name
    fprintf('Test Song: #%d = %s\n', k, songFiles(k).name);
    songData= spectogramCol(songPath);

    testSong = U'*songData;
    pval2 = w'*testSong;
    testClassfications(k) = pval2;

    if ((pval2 >= th1) && (pval2 <= th2))
        counterJazz = counterJazz+1;
    end
end
 percentCorrectJazz = 100*(counterJazz/10);

%Metal test data - 10 songs
songData = [];
counterMetal = 0;
for k = 1:10
    %this gets the file path of the song
```

```
    songPath = strcat(songFiles(k).folder, "/",
songFiles(k).name);

    %PRINTS song name
    fprintf('Test Song: #%d = %s\n', k, songFiles(k).name);
    songData= spectogramCol(songPath);

    testSong = U'*songData;
    pval3 = w'*testSong;
    testClassfications(k) = pval3;

    if (pval3 >= th2)
        counterMetal = counterMetal+1;
    end
end
 percentCorrectMetal = 100*(counterMetal/10);
 totalPercentCorrect =
100*((counterJazz+counterMetal+counterHiphop)/length(songFiles))
;

%% Print out Percent Accuracy
fprintf('\n');
fprintf('\n');
fprintf('Percent Accuracy for test Metal songs = %d \n',
percentCorrectMetal);
fprintf('Percent Accuracy for test Jazz songs = %d \n',
percentCorrectJazz);
fprintf('Percent Accuracy for test Hip Hop songs = %d \n',
percentCorrectHiphop);
fprintf('Total Percent Accuracy = %d \n', totalPercentCorrect);

%% Plot - projections onto w and mean(*)
figure(1)
%jazz = blue
plot(sortJazz,zeros(),'ob','Linewidth',2);
hold on
m1 = plot(meanJazz,0,'*b','Linewidth',10,'DisplayName','jazz
mean');

%hiphop = red
hold on
plot(sortHiphop,1,'or','Linewidth',2);
hold on
m2 =
plot(meanHiphop,1,'*r','Linewidth',10,'DisplayName','hiphop');

%metal = black
```

```
hold on
plot(sortMetal,ones()*2,'ok','Linewidth',2);
hold on
m3 =
plot(meanMetal,2,'*k','Linewidth',10,'DisplayName','metal');

%plot thresholds
hold on
m4 = xline(th1,'g','Linewidth',2);
hold on
xline(th2,'g','Linewidth',2);

ylim([0 5])
title('Test 3: 1-D Data Projection onto w');
ylabel('Arbitrary number');
xlabel('1-D projection onto w');
legend([m1,m2,m3,m4],'Jazz mean','Hiphop mean','Metal
mean','thresholds')


%% -------- DEFINING FUNCTIONS ------

%Spectogram function. Takes in data and computs the spectrogram,
returns
%the data reshaped into one column vector.
function data = spectogramCol(songPath)

        %geting music / spectogram data --> Change song files?
        [y,Fs] = audioread(songPath);

         %Lower sampling rate by a factor of 10
         Fs = Fs/10;
         song = y(1:Fs:5,:);
         S = song'; %Signal S equal to the song data amplitude
        L=5;
        time=length(song)/Fs; % total recorded time in seconds
        %time=length(L)/Fs; % total recorded time in seconds
        %t = (1:L)/Fs; % vector of time in seconds

        n=Fs*L; % n is the number of data points in signal
        t = linspace(0,L,n);


        k=(1/L)*[0:n/2-1 -n/2:-1]; %EVEN # of frequencys in Hz
        ks=fftshift(k);

        %Take the Spectogram of the song
```

```
        tslide=0:0.1:L; %Descrete time vector
        Sgt_spec = zeros(length(tslide),n);

        %gabor filter width
        a = 150;

        for jj=1:length(tslide)
            %applying gabor filter
            g=exp(-a*(t-tslide(jj)).^2);
            Sg=g.*S;
            Sgt=fft(Sg);%fourier transforming to the freuqency
domain

            %storing all the filtered signal at that time in the
spectogram vector
            Sgt_spec(jj,:) = fftshift(abs(Sgt));

        end

        data = reshape(Sgt_spec,
(length(tslide)*length(Sgt_spec)), 1);
end


%Trainer fucntion
%is used on the training data and performs the SVD and LDA
%also computs the first threshold (th1) and second threshold
(th2)
function [U,S,V,th1,th2,w,sortJazz,sortHiphop,sortMetal,
musicProjection] = trainer(jazz0,hiphop0,metal0,feature)


    nj = size(jazz0,2); nh = size(hiphop0,2); nm =
size(metal0,2);
    allMusic = [jazz0 hiphop0 metal0];

    %SVD
    [U,S,V] = svd(allMusic,'econ');

    % PCA projection onto principal components
    musicProjection = S*V';
    U = U(:,1:feature);
    Jazz = musicProjection(1:feature,1:nj);
    Hiphop = musicProjection(1:feature,nj+1:nj+nh);
    Metal = musicProjection(1:feature,nh+nm+1:nh+nm+nj);

    mj = mean(Jazz,2);
```

```
mh = mean(Hiphop,2);
mm = mean(Metal,2);
mTotal = mean(allMusic(1:feature),2);

% LDA -- WITHIN CLASS VARIANCES
Sw = 0;
for k=1:nj
    Sw = Sw + (Jazz(:,k)-mj)*(Jazz(:,k)-mj)';
end
for k=1:nh
    Sw = Sw + (Hiphop(:,k)-mh)*(Hiphop(:,k)-mh)';
end
for k=1:nm
    Sw = Sw + (Metal(:,k)-mm)*(Metal(:,k)-mm)';
end

% BETWEEN CLASS VARIANCES
SbJazz = (mj-mTotal)*(mj-mTotal)';
SbHiphop = (mh - mTotal)*(mh - mTotal)';
SbMetal = (mm - mTotal)*(mh - mTotal)';
Sb = SbJazz + SbHiphop + SbMetal;

% linear discriminant analysis
[V2,D] = eig(Sb,Sw);
[~,ind] = max(abs(diag(D)));
w = V2(:,ind); w = w/norm(w,2);

vjazz = w'*Jazz;
vhiphop = w'*Hiphop;
vmetal = w'*Metal;

sortJazz = sort(vjazz);
sortHiphop = sort(vhiphop);
sortMetal = sort(vmetal);

%THRESHOLDS
%finding threshold 1 - between metal and jazz
t1 = length(sortHiphop);
t2 = 1;
while sortHiphop(t1)>sortJazz(t2)
    t1 = t1-1;
    t2 = t2+1;
end
%threshold 1
th1 = (sortHiphop(t1)+sortJazz(t2))/2;

%finding threshold 2 - between hiphop and metal
```

```
    t1 = length(sortJazz);
    t2 = 1;
    while sortJazz(t1)>sortMetal(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    %threshold 2
    th2 = (sortJazz(t1)+sortMetal(t2))/2;

end
```