AMATH 482 HW3: Music Genre Identification

Ken Lo

March 8, 2018

Abstract

Use machine learning, SVD and Short-Time Fourier Transform to train a model to identify a 5-second music clip's genre and/or band, by training a model using a variety of 5-second music clips, labeled with their respective genres and/or bands.

1 Introduction and Overview

For us humans, soon upon hearing a short portion of a music, we can quickly identify the genre of the music, whether it is jazz, classical, rock, etc. However, how our brain perceives sound and translate frequency information into music is still uncertain to scientists. One way to identify music genre is to utilize how frequencies are distributed within a short time frame.

In this assignment, we will use the above approach - breaking down audio frequencies into time windows (Short-Time Fourier Transform), and use these information of 5-second music clips by different bands of different genres, and see if our model can accurately classify any 5-second clips' genres.

2 Theoretical Background

2.1 Sigular Value Decomposition (SVD)

A Singular Value Decomposition (SVD) takes the form

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$$

where

$$\begin{aligned}\mathbf{U} &\in \mathbb{C}^{m\times m}\text{is unitary}\\ \mathbf{V} &\in \mathbb{C}^{n\times n}\text{is unitary}\\ \mathbf{\Sigma} &\in \mathbb{R}^{m\times n}\text{is diagonal}\end{aligned}$$

Here the matrix $\mathbf{\Sigma}$ is diagonal with entries $\sigma_1 < \sigma_2 < \cdots < \sigma_n$.

By theorem 14.1.77 from class notes, any matrix $\mathbf{A}$ always has an SVD. So we can use this fact to perform SVD on our data matrix, in which the columns represent each 5-second audio clip. By doing so, our data matrix $\mathbf{A}$ will be diagonalized using basis $\mathbf{U}$ and $\mathbf{V}$, with singular values in $\mathbf{\Sigma}$. Then, the proper orthogonal modes will take place as vectors in $\mathbf{U}$, and the singular values will tell us how significant these POMs are in representing the audio clips.

Our goal with SVD is to generalize and create a lower dimensional space to represent our audio clips, then we can use any sample clip's projection onto the space to predict and classify which genre or band the given clip belongs to.

2.2 Fourier Transform

The Fourier Transform is an integral transform defined over $x \in [-\infty, \infty]$, given by the following equation:

$$F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp(-ikx) f(x) dx \tag{1}$$

The Transform decomposes a function (a signal over time) into a complex-valued function of frequency, where its real component represents the amount(or power) of frequency present, and the complex component represents the phase offset of the sinusoidal wave in such frequency. However, a limitation to the Fourier Transform is that it removes all the time-related information regarding a signal, we will discuss how Gabor Transform can remedy such limitation in the next subsection.

2.3 Gábor Transform (Short-time Fourier Transform)

In order to localize both time and frequency of a signal, Gábor Dénes introduced the kernel

$$g_{t,w}(\tau) = \exp(i\omega t) g(\tau - t) \tag{2}$$

where $g(\tau - t)$ was introduced to localize both time and frequency. As a result, the complete Gábor Transform (or short-time Fourier transform) is

$$\bar{f}_g(t, k) = \int_{-\infty}^{\infty} f(\tau) \bar{g}(\tau - t) \exp(-ik\tau) d\tau = (f, g_{t,k}) \tag{3}$$

With this transformation, we then can transform our signal (audio clips) into frequency and time information for our classifying task.

2.4 Naive Bayes Classifier

The Naive Bayes Classifier follows a conditional probability model, where

$$p(C_k|x) = \frac{p(C_k) p(x|C_k)}{p(x)} \tag{4}$$

Then the classifier is

$$\hat{y} = \underset{k \in (1,\dots,K)}{\operatorname{argmax}} \, p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{5}$$

In words, the naive bayes classifier treats each features to *independently* contribute to the probability that the sample a is in category 1. When we ask a trained naive bayes model to classify a test sample, it will compute the probabilities that the test sample is in whichever category, and classify it the category with the highest probability.

3 Algorithm Implementation and Development

3.1 Gathering Data

The data used in this assignment is obtained by downloading audios from YouTube, then converted them to .wav format. This was done using Python.

3.2 Getting 5-second clips

Having downloaded the longer audios files from YouTube (which are generally a collection of songs of bands). We randomly (by using a random number generator in MATLAB) chose locations in the audio files where we would cut out 5-second audio clips.

3.3 Spectrogram and Short Time Fourier Transform

To make time-frequency data usable, we used MATLAB to obtain the spectrograms of our data, by using the `spectrogram()` method, which uses short-time fourier transform.

By default, the `spectrogram()` method splits the time-frequency signal into 8 windows with equal time, and these is a 50% overlap between windows.

3.4 Singular Value Decomposition

With the spectrograms representing each 5-second audio clips, we then perform SVD on these data to determine some dominant proper orthogonal modes with which we can create a lower dimensional representation of the data.

3.5 Training Naive Bayes

We then selected a number of the more dominant proper orthogonal modes, and trained a Naive Bayes Classifier, using the 5-second clips' projection onto the POMs as features, and their respective genres/bands as labels.

In order to avoid over-fitting, we splitted the data into 80% for training and 20% for testing, which would give us more confidence of our trained model's accuracy. This process is repeated multiple times, using different random subsets for training and testing, to achieve cross-validation.

4 Computational Results

4.1 Test 1: Band Classification

For Test 1, I chose three different bands of different genres, which are Miles Davis (Jazz), Ice Cube (Rap) and Chopin(Classical).

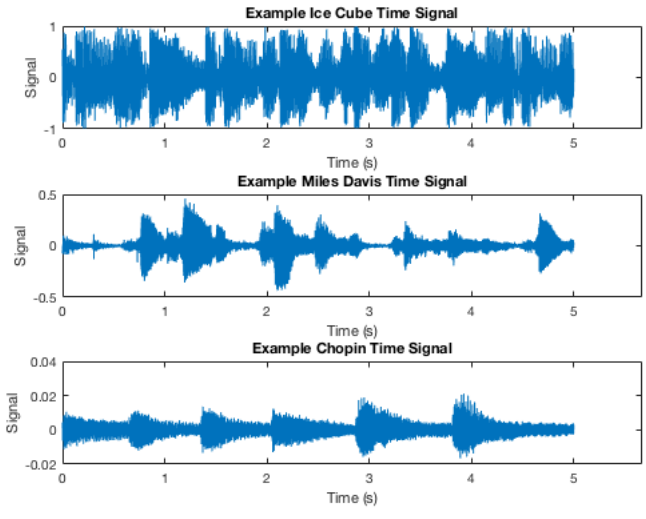

Figure 1: Example of the time-frequency of 5-second clips of Ice Cube, Miles Davis and Chopin

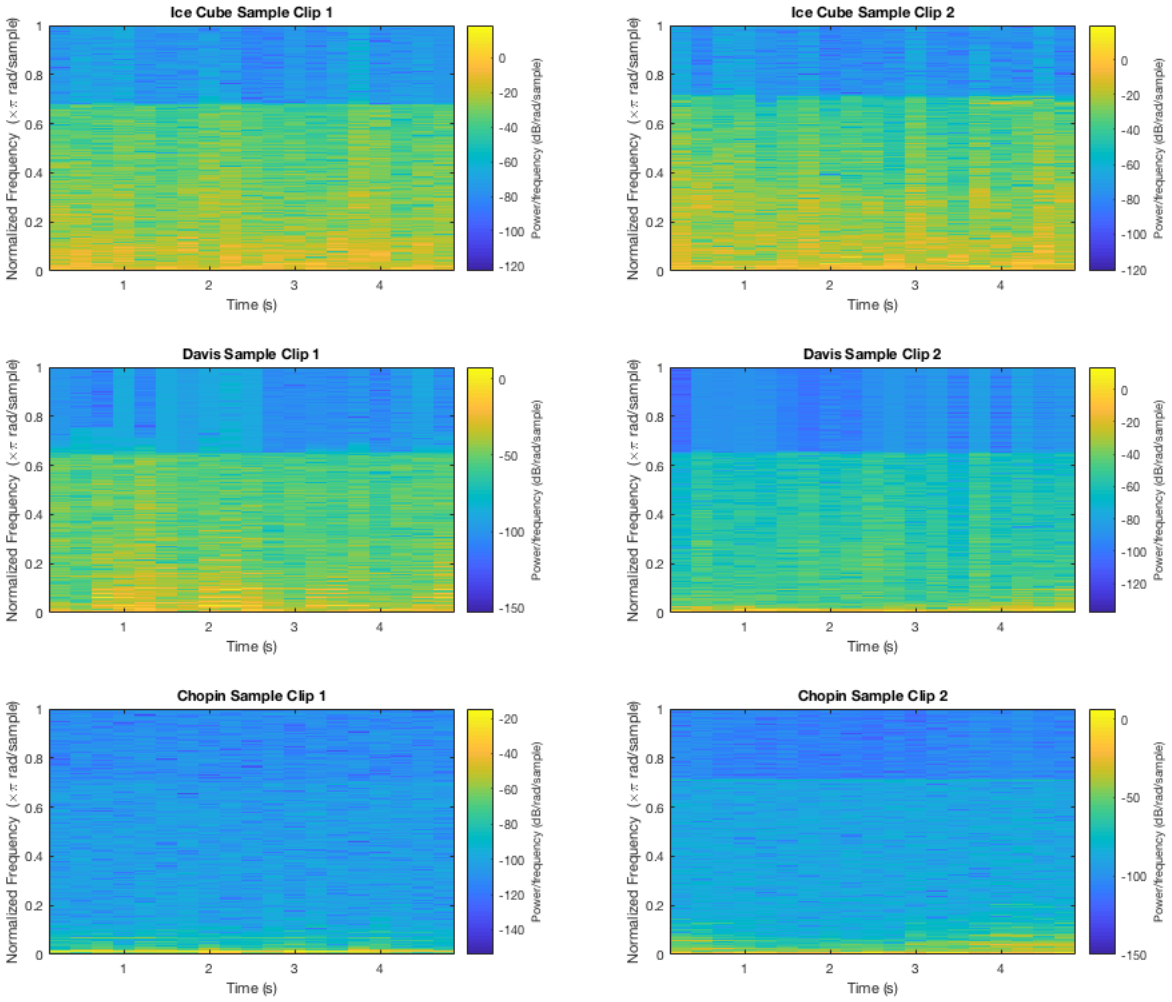

Figure 2: Example spectrograms from each of the band's two music

From figure 1, we could see distinct differences from each of their music, where Ice Cube's music is more "explosive", and Davis' and Chopin's music are relatively "calm". From figure 2, we can see the spectrograms of two audio clips from each of the band. We can easily observe that Ice Cube's spectrogram has more yellow and green in it, Miles Davis' spectrogram is less yellow and has slightly more blue, while Chopin's spectrograms has the most blue colors out of all spectrograms. Here, the colors represent the power(amont) of frequency existed in that time-window, where warmer color (yellow) represents a stronger presence, and colder color (blue - dark blue) means at that time-window the corresponding frequency's presence is minimal.

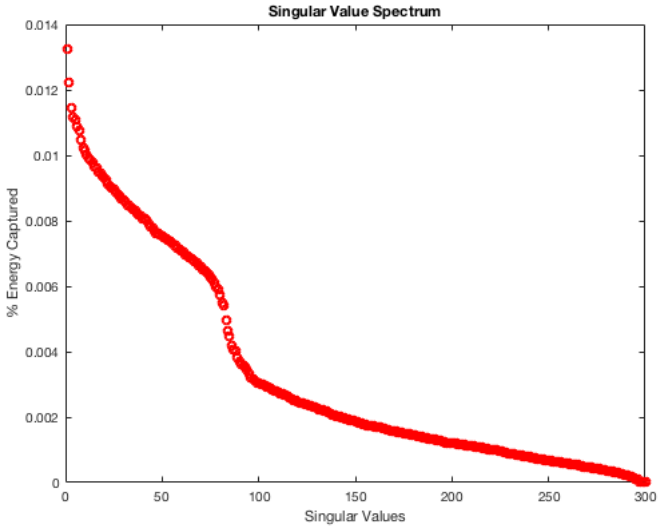

Figure 3: Singular Value Spectrum of the three bands

Figure 3 shows the singular value spectrum of the three bands, we can see that despite there is a few modes that are more dominant than the others, each of those modes only takes up of about 1% of all energy.

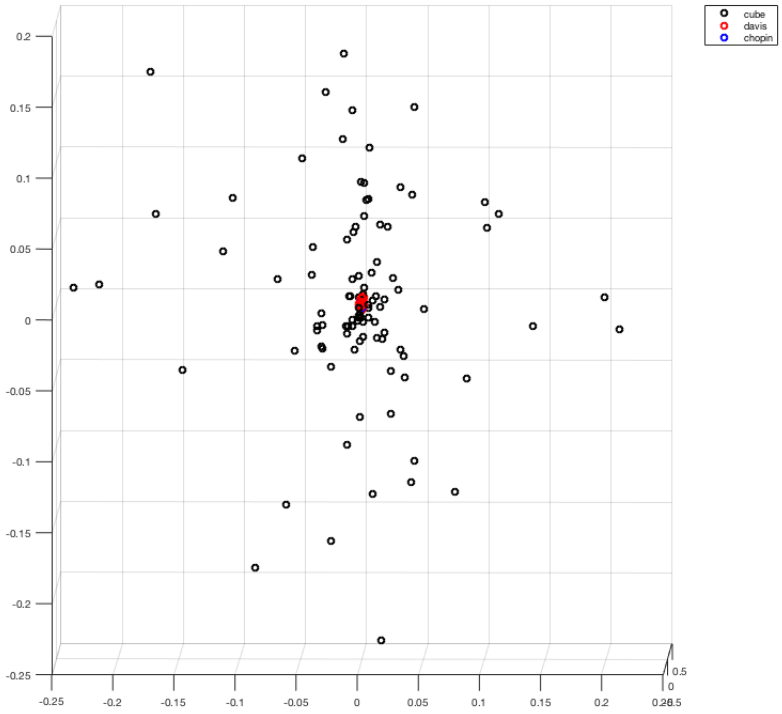

Figure 4: projection of the audio clips onto the first three modes

From figure 4 we can see Ice Cube's music (black) are mostly outside of Miles Davis' music (red) which wraps around Chopin's music(blue). Here we can guess that we could separate these three bands by drawing "circles" around each band's dots.

For classification, I then used a Naive Bayes Classifier, using the projections onto the first 100 modes as features.

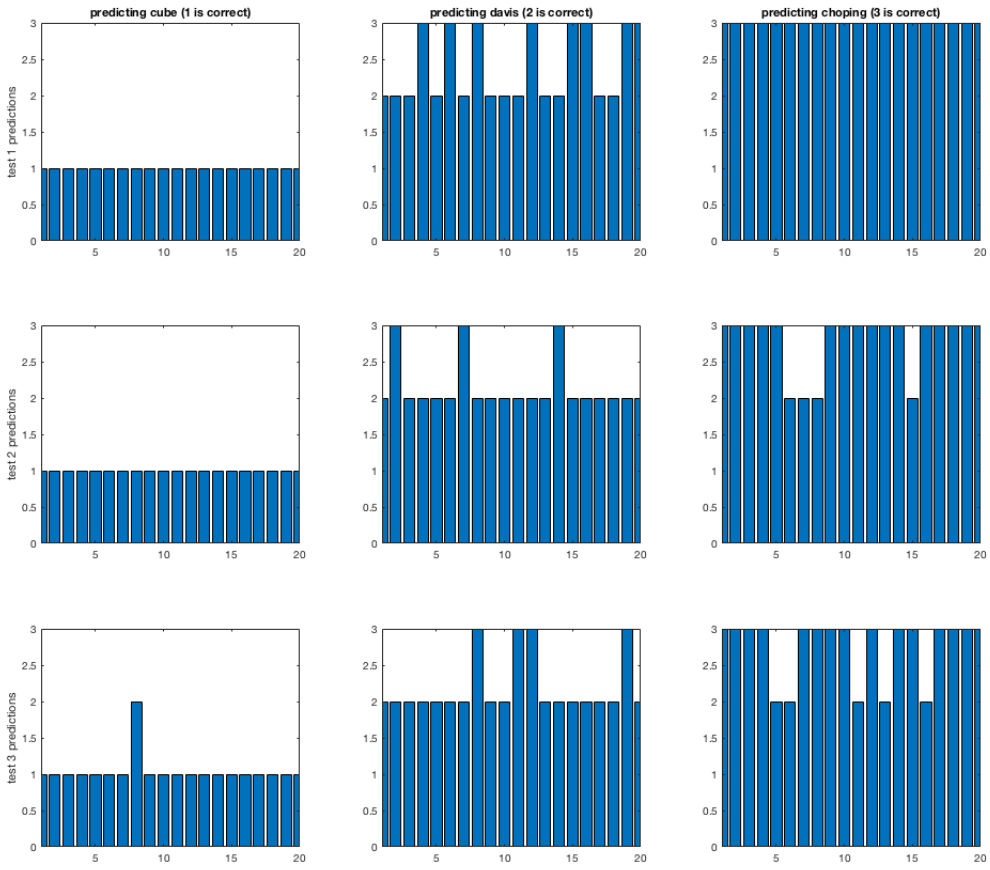

Figure 5: Naive Bayes Classification results, each row represents each trial using different random samples, and each column represents the correctness of predicting each band.

From figure 5 we see that our classifier performs quite well in classifying test audio clips. On average over the three trials, the classifier only misclassified about 10 test clips, out of 60. This translates to about 83% accuracy.

4.2 Test 2: The Case for Seattle (Classical)

For test 2, I chose Beethovan, Chopin and Mozart from the classical genres to see if the computer could distinguish between their music.

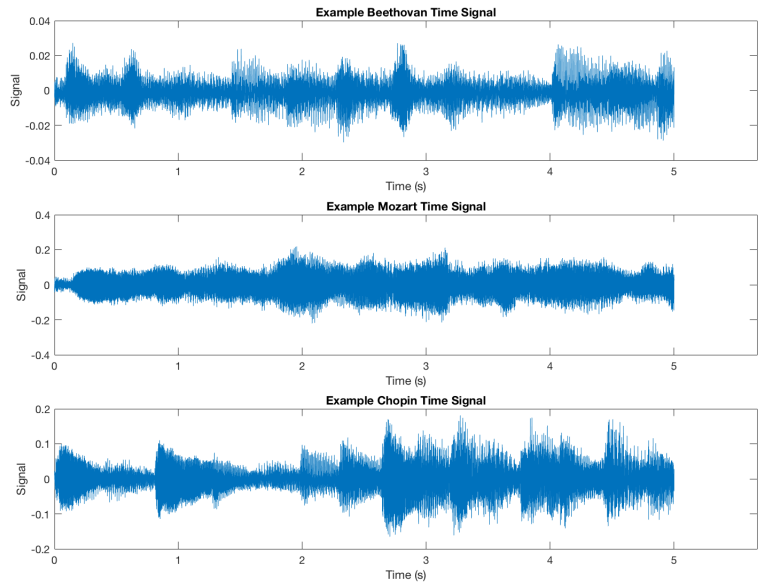

Figure 6: Example Time Frequency plots of audio clips by Chopin, Mozart and Beethovan

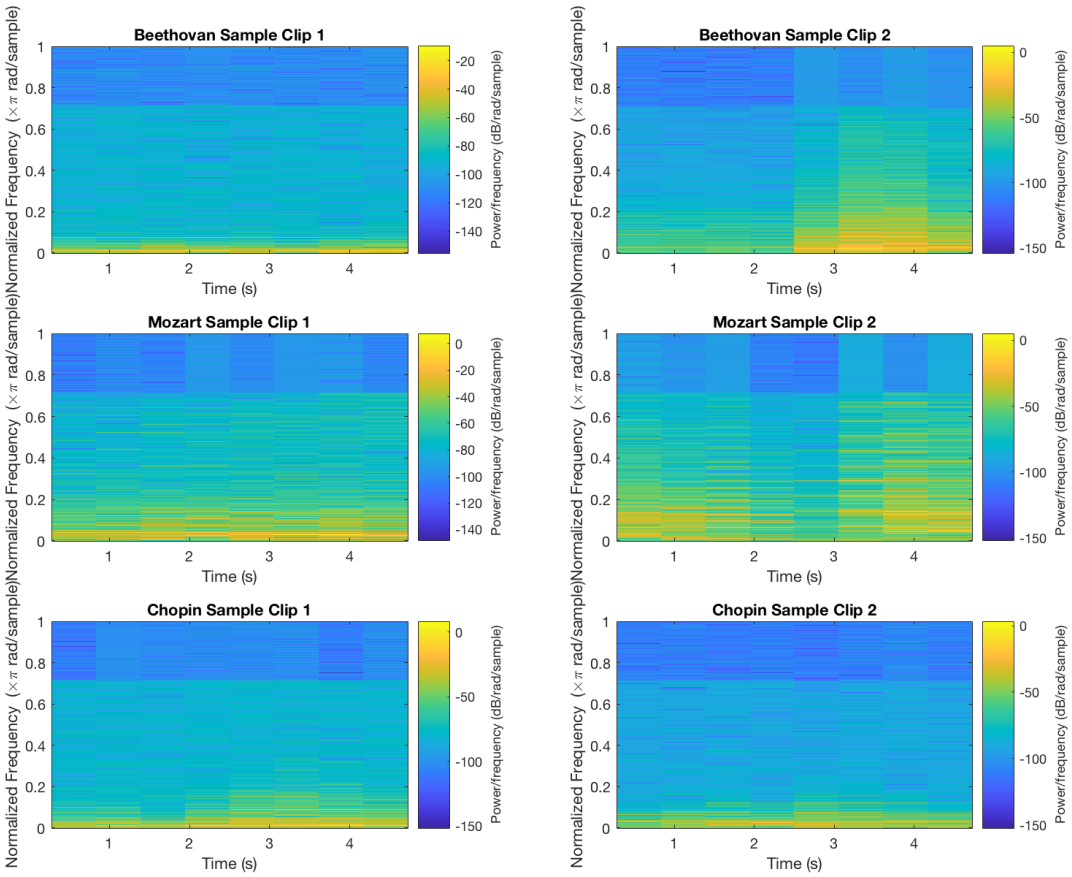

Figure 7: Example Spectrograms from each musician

From figures 6 and 7, we could see that each of the musician's audio clips looked very similar, in both the time frequency plots and the spectrograms.

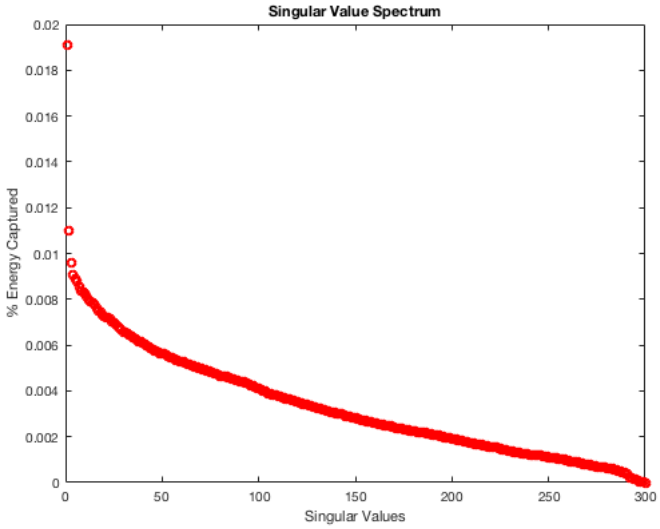

Figure 8: Singular Value Spectrum of the three classical musicians

From the Singular Value Spectrum in figure 8, we could see there the is one more apparent dominant mode which has about 1.9% of the energy, with a few less but still dominant modes that have roughly 1% of the energy.

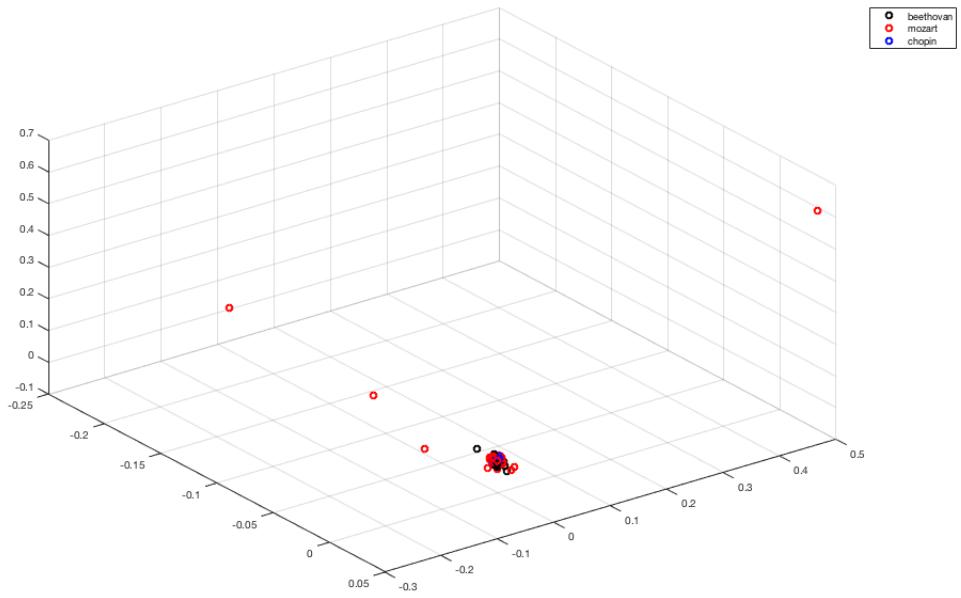

Figure 9: Projection of the three musicians onto the first three modes

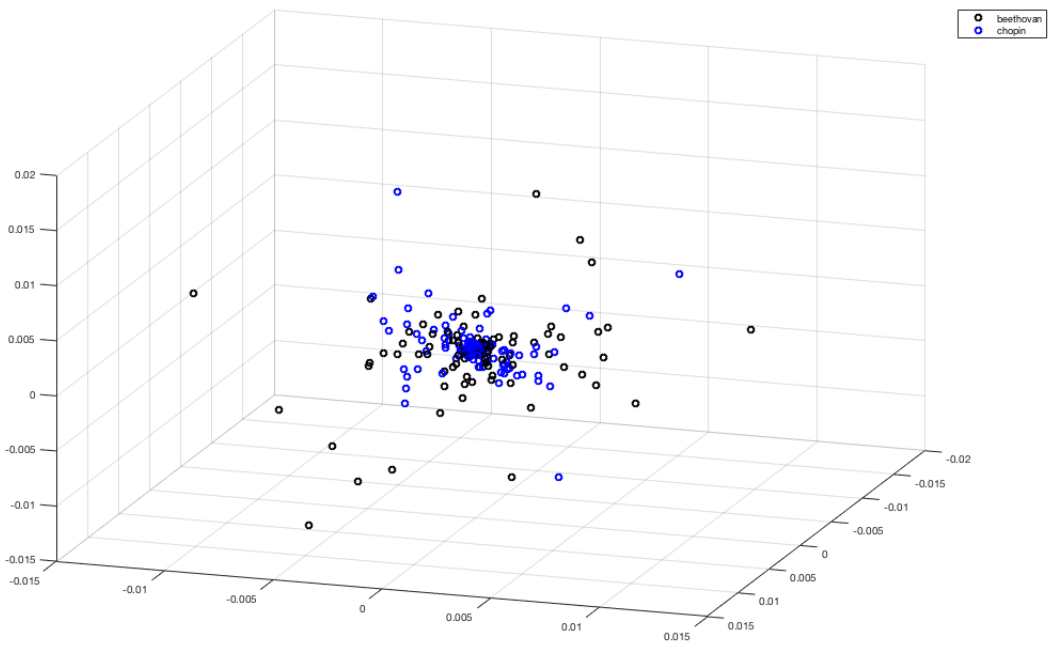

Figure 10: Projection of only Beethovan and Chopin onto the first three modes

Figure 9 shows that the three musicians are clustered together when projected onto the first three most dominant modes. When we take away the red circles (Mozart), as shown in figure 10, we could still see that the blue and red circles are still somewhat clustered together. This indicates that it would be more difficult to separate them and accurately classify a test clip.

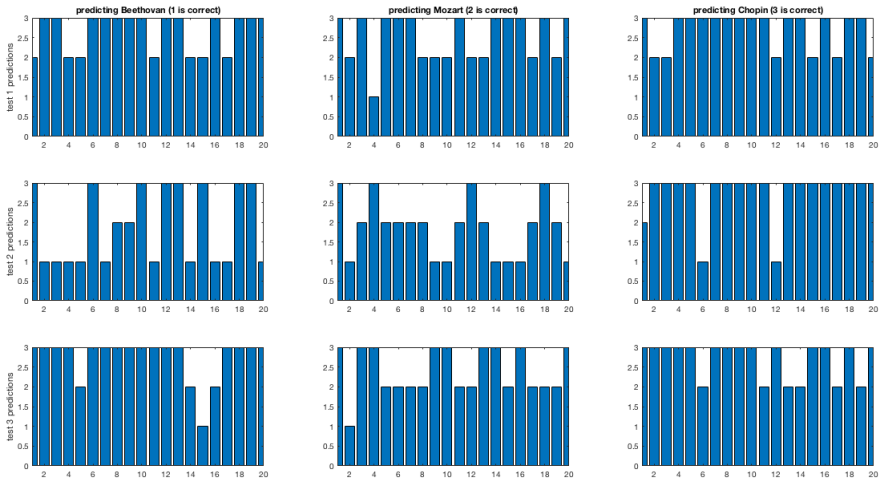

Figure 11: Naive Bayes Classification results, where each row represents each trial with different train and test samples, and each column represents the correctness of the classification.

As confirmed by the results shown in figure 10, the naive bayes classifier, using the first 100 proper orthogonal modes, failed to correctly distinguish Beethovan's music from the other two musicians, and the overall accuracy from the three trials range from 38% - 60%.

4.3 Test 3: Genre Classification

For test 3, I chose Miles Davis, John Coltrance and Glen Miller to represent the Jazz Genre; Mozart, Beethovan and Chopin to represent the classical genre; Ice Cube, Dr. Dre and Dame D.O.L.L.A to represent the rap genre.

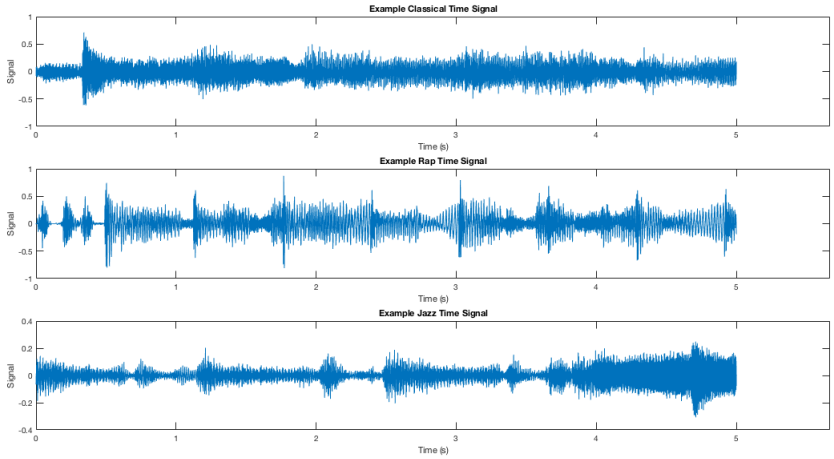

Figure 12: Example time-frequency plot of a sample audio clip from each genre.

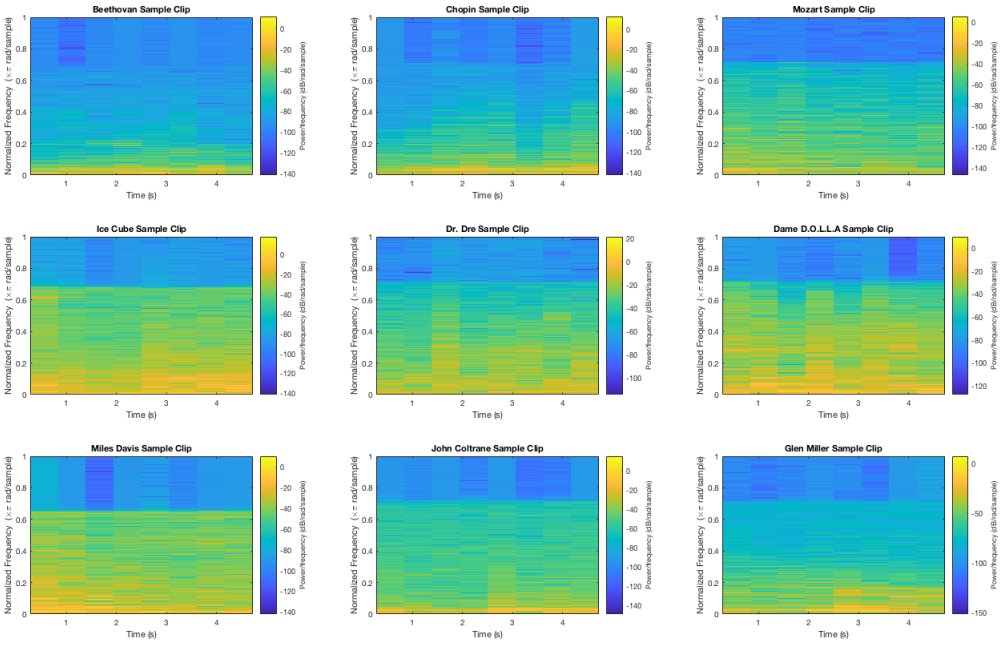

Figure 13: Spectrograms of two example clips from each genre.

Just like test 1 and test 2, the spectrograms in figure 13 show similarities within each genres and differences among different genres.

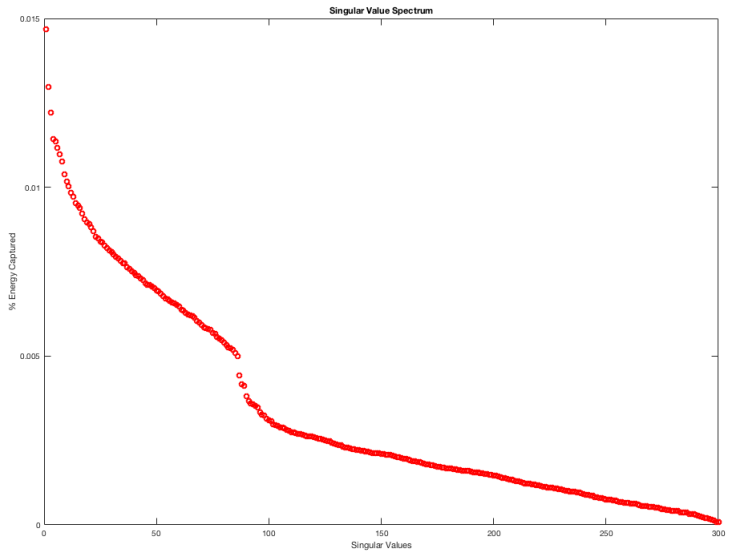

Figure 14: Singular Value Spectrum of the three genres.

Compare to the singular value spectrums in tests 1 and 2 (see figure 3 and 8), we can see from figure 14 that the singular values are more separated, meaning that the SVD was able to find more commonalities between music. This is expected as music from the same genres will likely have similar spectrograms.

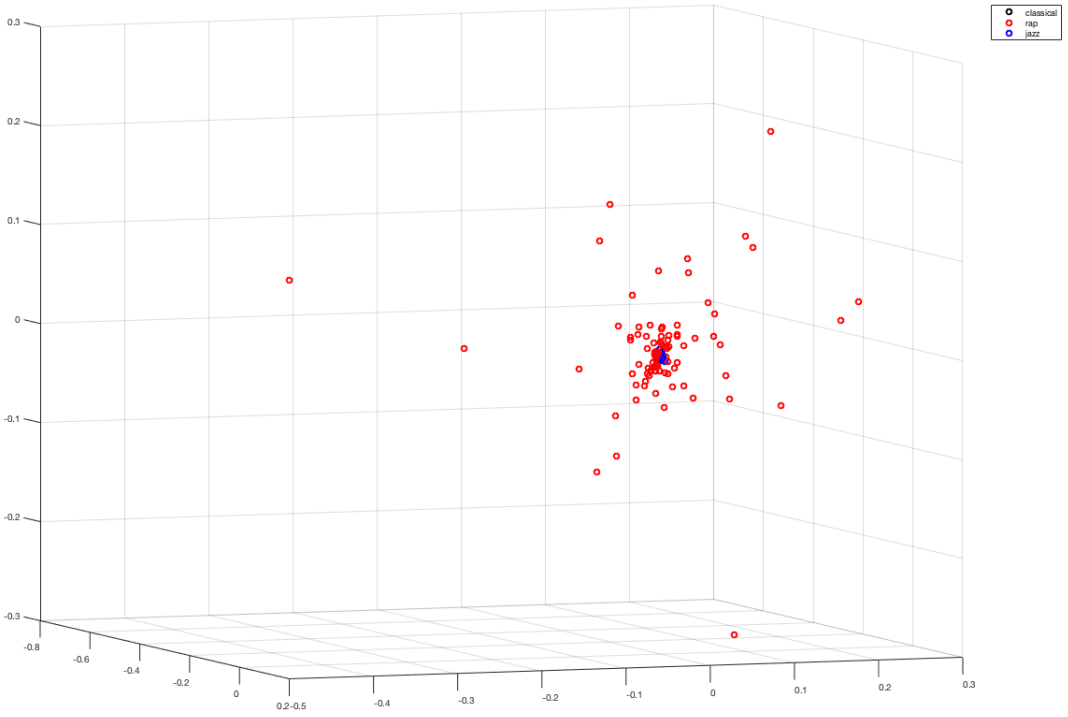

Figure 15: Projections of the audio clips from different genres onto the first three modes

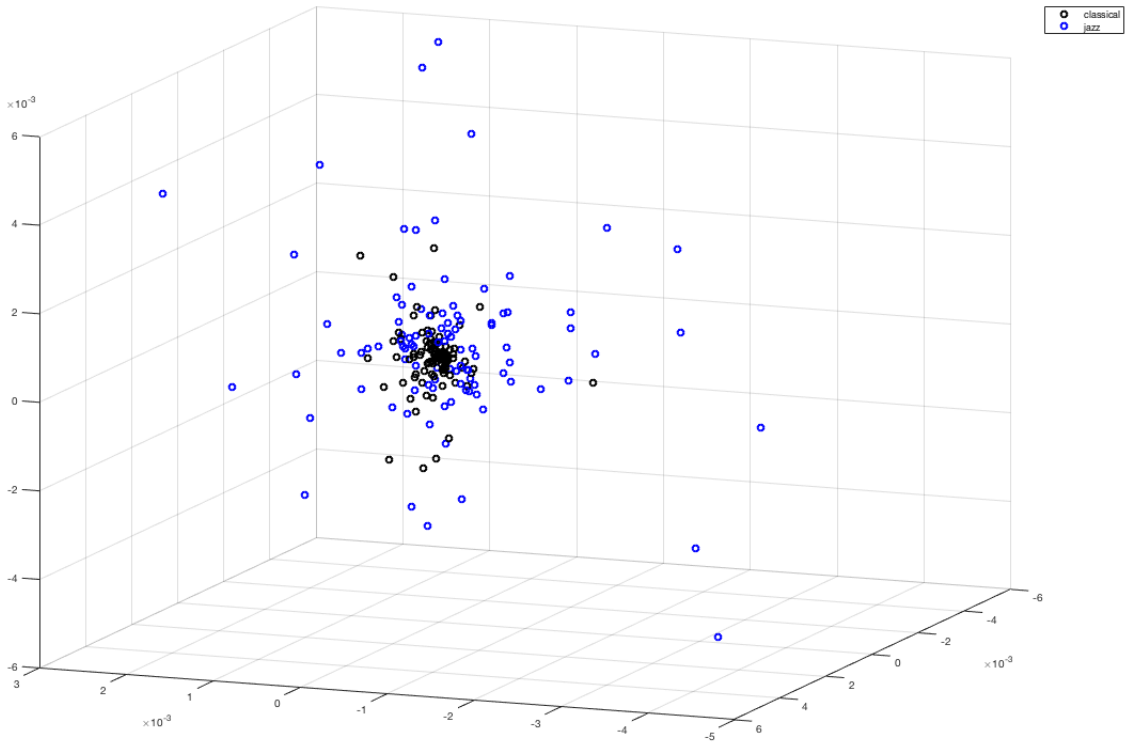

Figure 16: Projections of classical and jazz music onto the first three modes

In figure 15 and 16, the projects of the genres onto the first three modes resembles figure 4, where we only had one bands per genre, that rap music (red) wraps around jazz(blue) music which wraps around classical music (black). (Note that in figure 4 rap music is represented by black circles, jazz is represented by red, and classical is represented by blue instead). So, we should expect similar, if not better, performance from our classifier.

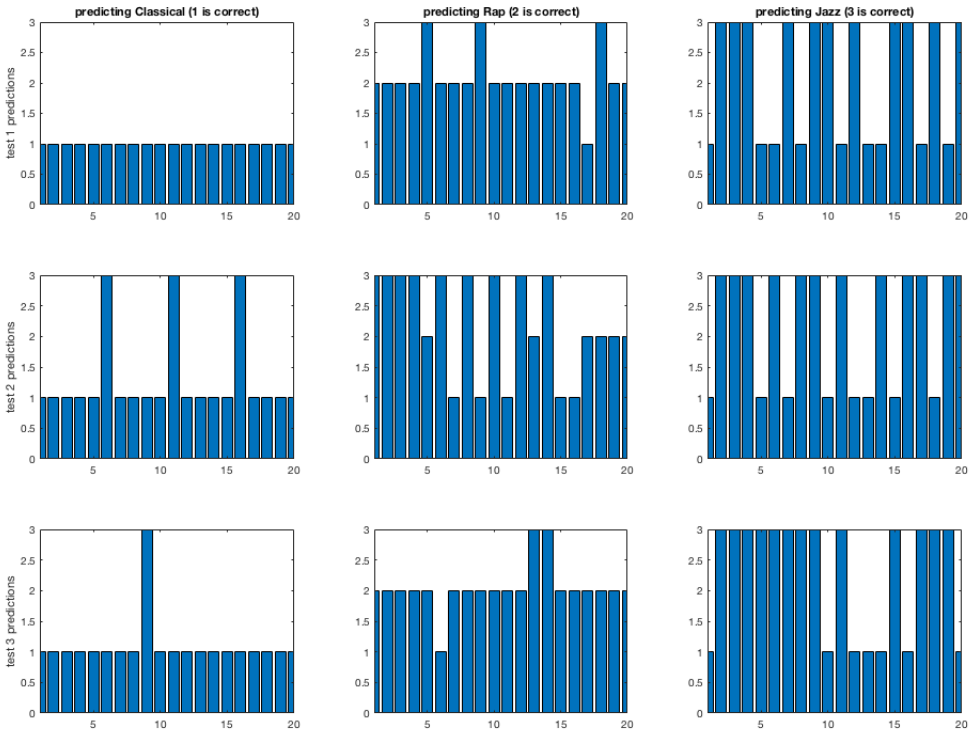

Figure 17: Naive Bayes Classification results, where each row represents each trial with different train and test samples, and each column represents the correctness of the classification of different genres.

Using the first 100 proper orthogonal modes to train the Naive Bayes Classifier, the accuracy ranges from 58% - 80% using different training and testing samples. This is illustrated in figure 17.

5 Summary and Conclusions

Through this assignment, I experienced how to turn an audio signal into time and frequency information. With this information, I have seen how SVD could give us a low-dimensional representation of our data, with which we can train a machine learning model to identify music genres.

Our classifier only gave us a roughly 80% accuracy, however this can be attributed to the small sample size and the way we randomly sample our music. Since the sampled music might take place at different parts of the song (intro, chorus, etc.), it would make it more difficult to for our classifier to identify any arbitrary music clip. If we trained and tested our classifier using only a certain parts of the song, such as the chorus, a better accuracy would be much more likely to result in. In addition, using a larger training set size could also help our model better generalize the features that each genre possess.

A MATLAB functions used and brief implementation explanation

- audioread - read an audio file into time-frequency matrix
- audioinfo - returns meta data of an audio file, such as name, channels, frequency, sample rate, etc.
- spectrogram - perform short-time fourier transform on the time-frequency matrix and plot the spectrogram
- svd - perform svd on the matrix
- scatter3 - make a 3d scatterplot
- fitcnb - fit a naive bayes classifier
- predict - predict the classification results of test data
- randperm - returns a matrix with random numbers, used for randomly select data for training and testing the model

B MATLAB codes

B.1 train_test_split.m

```
% split data into train and test sets
function [xtrain,ytrain,xtest,ytest] = train_test_split(v)
    frommode = 1;
    tomode = 3;
    q1 = randperm(100);
    q2 = randperm(100);
    q3 = randperm(100);

    xice = v(1:100, frommode:tomode);
    xmd = v(101:200, frommode:tomode);
    xcho = v(201:300, frommode:tomode);
    xtrain = [xice(q1(1:80),:); xmd(q2(1:80),:); xcho(q3(1:80),:)];
    xtest = [xice(q1(81:end),:); xmd(q2(81:end),:); xcho(q3(81:end),:)];
    ytrain = zeros(240,1);
    for i=1:240
        if i <= 80
            ytrain(i) = 1;
        elseif i <= 160
                ytrain(i) = 2;
        else
            ytrain(i) = 3;
        end
    end
    ytest = zeros(60,1);
    for i=1:60
        if i<= 20
            ytest(i) = 1;
        elseif i <= 40
            ytest(i) = 2;
        else
            ytest(i) = 3;
```

```
        end
    end
end
```

B.2 get_random_sample.m

```
% Get 5-second clips randomly from the given FILE
function A = get_random_sample(FILENAME, numSamples)
    A = [];
    finfo = audioinfo(FILENAME);
    duration = finfo.Duration;
    sr = finfo.SampleRate;
    R = rand(1,numSamples);
    samp_starttime = R * (duration - 5);
    for i=1:size(samp_starttime, 2)
       [y, fs] = audioread(FILENAME, [round(samp_starttime(i) * sr), round((samp_starttim
       yMono = sum(y,2) / size(y, 2);
       A = [A yMono];
    end
end
```

B.3 test1.m

```
%% load data
% FILEPATHS:
% 'data/clean/md-collection.wav'
% 'data/clean/cho-collection.wav'
% 'data/clean/ice-collection.wav'

md = get_random_sample('data/clean/md-collection.wav', 100);
cho = get_random_sample('data/clean/cho-collection.wav', 100);
ice = get_random_sample('data/clean/ice-collection.wav', 100);
ss = size(md, 1) / 5; % samples per second

%% plot time signal
figure
subplot(3,1,1)
plot(ice(:,1))
title ('Example Ice Cube Time Signal')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
ylabel('Signal')
subplot(3,1,2)
plot(md(:,1))
title('Example Miles Davis Time Signal')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
ylabel('Signal')
subplot(3,1,3)
plot(cho(:,1))
title('Example Chopin Time Signal')
```

```
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
ylabel('Signal')

%% plot example spectrograms
wlength = ss * 0.5; %% window length for STFT
figure
subplot(3,2,1)
spectrogram(ice(:,1),wlength,'yaxis')
title('Ice Cube Sample Clip 1')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
subplot(3,2,2)
spectrogram(ice(:,2),wlength, 'yaxis')
title('Ice Cube Sample Clip 2')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
subplot(3,2,3)
spectrogram(md(:,1),wlength, 'yaxis')
title('Davis Sample Clip 1')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
subplot(3,2,4)
spectrogram(md(:,2),wlength, 'yaxis')
title('Davis Sample Clip 2')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
subplot(3,2,5)
spectrogram(cho(:,1),wlength, 'yaxis')
title('Chopin Sample Clip 1')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
subplot(3,2,6)
spectrogram(cho(:,2),wlength, 'yaxis')
title('Chopin Sample Clip 2')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')


%% stack all three data matrices and get spectrogram for each clip
Dall = [ice, md, cho];
% rap: 1-100, jazz: 101-200, clas: 201-300
Sall = [];
for i=1:size(Dall,2)
    %sp = spectrogram(Dall(:, i), round(wlength));
    sp = spectrogram(Dall(:, i));
```

```
    Sall = [Sall, sp(:)];
end


%% svd
[u,s,v] = svd(Sall, 'econ');

%% plot singular value spectrum
figure
plot(diag(s)/sum(diag(s)), 'ro', 'Linewidth', 2)
title('Singular Value Spectrum')
ylabel('% Energy Captured')
xlabel('Singular Values')

%%
figure
scatter3(v(1:100,1), v(1:100,2), v(1:100,3), 'ko', 'Linewidth', 2)
hold on
scatter3(v(101:200,1), v(101:200,2), v(101:200,3), 'ro', 'Linewidth', 2)
scatter3(v(201:300,1), v(201:300,2), v(201:300,3), 'bo', 'Linewidth', 2)
legend('cube', 'davis', 'chopin')
%%
% frommode = 1;
% tomode = 100;
% q1 = randperm(100);
% q2 = randperm(100);
% q3 = randperm(100);
%
% xice = v(1:100, frommode:tomode);
% xmd = v(101:200, frommode:tomode);
% xcho = v(201:300, frommode:tomode);
% xtrain = [xice(q1(1:80),:); xmd(q2(1:80),:); xcho(q3(1:80),:)];
% xtest = [xice(q1(81:end),:); xmd(q2(81:end),:); xcho(q3(81:end),:)];
% ytrain = zeros(240,1);
% for i=1:240
%     if i <= 80
%         ytrain(i) = 1;
%     elseif i <= 160
%             ytrain(i) = 2;
%     else
%         ytrain(i) = 3;
%     end
% end

[xtrain, ytrain, xtest, ytest] = train_test_split(v);
classifier = fitcnb(real(xtrain), ytrain);
predicted = predict(classifier, real(xtest));
figure
subplot(3,3,1)
bar(predicted(1:20))
axis([1 20 0 3])
title('predicting cube (1 is correct)')
ylabel('test 1 predictions')
```

```
subplot(3,3,2)
bar(predicted(21:40))
axis([1 20 0 3])
title('predicting davis (2 is correct)')
subplot(3,3,3)
bar(predicted(41:60))
axis([1 20 0 3])
title('predicting choping (3 is correct)')

[xtrain, ytrain, xtest, ytest] = train_test_split(v);
classifier = fitcnb(real(xtrain), ytrain);
predicted = predict(classifier, real(xtest));
subplot(3,3,4)
bar(predicted(1:20))
ylabel('test 2 predictions')
axis([1 20 0 3])
subplot(3,3,5)
bar(predicted(21:40))
axis([1 20 0 3])
subplot(3,3,6)
bar(predicted(41:60))
axis([1 20 0 3])

[xtrain, ytrain, xtest, ytest] = train_test_split(v);
classifier = fitcecoc(real(xtrain), ytrain);
predicted = predict(classifier, real(xtest));
subplot(3,3,7)
bar(predicted(1:20))
axis([1 20 0 3])
ylabel('test 3 predictions')
subplot(3,3,8)
bar(predicted(21:40))
axis([1 20 0 3])
subplot(3,3,9)
bar(predicted(41:60))
axis([1 20 0 3])
```

B.4 test2.m

```
%% load data
% FILEPATHS:
% 'data/clean/mj-collection.wav'
% 'data/clean/bee-collection.wav'
% 'data/clean/ice-collection.wav'

moz = get_random_sample('data/clean/moz-collection.wav', 100);
cho = get_random_sample('data/clean/cho-collection.wav', 100);
bee = get_random_sample('data/clean/bee-collection.wav', 100);
ss = size(moz, 1) / 5; % samples per second

%% plot time signal
figure
subplot(3,1,1)
```

```
plot(bee(:,1))
title ('Example Beethovan Time Signal')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
ylabel('Signal')
subplot(3,1,2)
plot(moz(:,1))
title('Example Mozart Time Signal')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
ylabel('Signal')
subplot(3,1,3)
plot(cho(:,1))
title('Example Chopin Time Signal')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
ylabel('Signal')

%% plot example spectrograms
%wlength = ss * 0.5; %% window length for STFT
figure
subplot(3,2,1)
spectrogram(bee(:,1),'yaxis')
title('Beethovan Sample Clip 1')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
subplot(3,2,2)
spectrogram(bee(:,2), 'yaxis')
title('Beethovan Sample Clip 2')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
subplot(3,2,3)
spectrogram(moz(:,1), 'yaxis')
title('Mozart Sample Clip 1')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
subplot(3,2,4)
spectrogram(moz(:,2), 'yaxis')
title('Mozart Sample Clip 2')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
subplot(3,2,5)
spectrogram(cho(:,1), 'yaxis')
title('Chopin Sample Clip 1')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
```

```
xlabel('Time (s)')
subplot(3,2,6)
spectrogram(cho(:,2), 'yaxis')
title('Chopin Sample Clip 2')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')


%% stack all three data matrices and get spectrogram for each clip
Dall = [bee, moz, cho];
% beethovan: 1-100, mozart: 101-200, chopin: 201-300
Sall = [];
for i=1:size(Dall,2)
    %sp = spectrogram(Dall(:, i), round(wlength));
    sp = spectrogram(Dall(:, i));
    Sall = [Sall, sp(:)];
end


%% svd
[u,s,v] = svd(Sall, 'econ');

%% plot singular value spectrum
figure
plot(diag(s)/sum(diag(s)), 'ro', 'Linewidth', 2)
title('Singular Value Spectrum')
ylabel('% Energy Captured')
xlabel('Singular Values')

%%
figure
scatter3(v(1:100,1), v(1:100,2), v(1:100,3), 'ko', 'Linewidth', 2)
hold on
%scatter3(v(101:200,1), v(101:200,2), v(101:200,3), 'ro', 'Linewidth', 2)
scatter3(v(201:300,1), v(201:300,2), v(201:300,3), 'bo', 'Linewidth', 2)
legend('beethovan', 'chopin') %'mozart', 'chopin')
%%
figure
plot(v(1:100,1), v(1:100, 2), 'ko')
hold on
plot(v(101:200,1), v(101:200,2), 'ro')
plot(v(201:300,1), v(201:300,2), 'go')
%%
% frommode = 1;
% tomode = 100;
% q1 = randperm(100);
% q2 = randperm(100);
% q3 = randperm(100);
%
% xice = v(1:100, frommode:tomode);
% xmd = v(101:200, frommode:tomode);
% xcho = v(201:300, frommode:tomode);
```

```
% xtrain = [xice(q1(1:80),:); xmd(q2(1:80),:); xcho(q3(1:80),:)];
% xtest = [xice(q1(81:end),:); xmd(q2(81:end),:); xcho(q3(81:end),:)];
% ytrain = zeros(240,1);
% for i=1:240
%     if i <= 80
%         ytrain(i) = 1;
%     elseif i <= 160
%             ytrain(i) = 2;
%     else
%         ytrain(i) = 3;
%     end
% end

[xtrain, ytrain, xtest, ytest] = train_test_split(v);
%classifier = fitcnb(real(xtrain), ytrain);
classifier = fitcknn(real(xtrain), ytrain, 'NumNeighbors', 3);
predicted = predict(classifier, real(xtest));
figure
subplot(3,3,1)
bar(predicted(1:20))
axis([1 20 0 3])
title('predicting Beethovan (1 is correct)')
ylabel('test 1 predictions')
subplot(3,3,2)
bar(predicted(21:40))
axis([1 20 0 3])
title('predicting Mozart (2 is correct)')
subplot(3,3,3)
bar(predicted(41:60))
axis([1 20 0 3])
title('predicting Chopin (3 is correct)')

[xtrain, ytrain, xtest, ytest] = train_test_split(v);
%classifier = fitcnb(real(xtrain), ytrain);
classifier = fitcknn(real(xtrain), ytrain, 'NumNeighbors', 3);
predicted = predict(classifier, real(xtest));
subplot(3,3,4)
bar(predicted(1:20))
ylabel('test 2 predictions')
axis([1 20 0 3])
subplot(3,3,5)
bar(predicted(21:40))
axis([1 20 0 3])
subplot(3,3,6)
bar(predicted(41:60))
axis([1 20 0 3])

[xtrain, ytrain, xtest, ytest] = train_test_split(v);
%classifier = fitcknn(real(xtrain), ytrain, 'NumNeighbors', 5);
classifier = fitcknn(real(xtrain), ytrain, 'NumNeighbors', 3);
predicted = predict(classifier, real(xtest));
subplot(3,3,7)
bar(predicted(1:20))
```

```
axis([1 20 0 3])
ylabel('test 3 predictions')
subplot(3,3,8)
bar(predicted(21:40))
axis([1 20 0 3])
subplot(3,3,9)
bar(predicted(41:60))
axis([1 20 0 3])
```

B.5 test3.m

```
%% load data

mozart = get_random_sample('data/clean/moz-collection.wav', 34);
chopin = get_random_sample('data/clean/cho-collection.wav', 33);
beetho = get_random_sample('data/clean/bee-collection.wav', 33);
cube = get_random_sample('data/clean/ice-collection.wav', 34);
dre = get_random_sample('data/clean/dre-collection.wav', 33);
dame = get_random_sample('data/clean/dame-collection.wav',33);
davis = get_random_sample('data/clean/md-collection.wav',34);
coltrane = get_random_sample('data/clean/jc-collection.wav',33);
miller = get_random_sample('data/clean/gm-collection.wav',33);
ss = size(mozart, 1) / 5; % samples per second

%% plot time signal
figure
subplot(3,1,1)
plot(mozart(:,1))
title ('Example Classical (Mozart) Time Signal')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
ylabel('Signal')
subplot(3,1,2)
plot(cube(:,1))
title('Example Rap (Ice Cube) Time Signal')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
ylabel('Signal')
subplot(3,1,3)
plot(davis(:,1))
title('Example Jazz (Miles Davis) Time Signal')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
ylabel('Signal')

%% plot example spectrograms
%wlength = ss * 0.5; %% window length for STFT
figure
subplot(3,3,1)
spectrogram(beetho(:,1),'yaxis')
```

```
title('Beethovan Sample Clip')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
subplot(3,3,2)
spectrogram(chopin(:,2), 'yaxis')
title('Chopin Sample Clip')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
subplot(3,3,3)
spectrogram(mozart(:,1), 'yaxis')
title('Mozart Sample Clip')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
subplot(3,3,4)
spectrogram(cube(:,1), 'yaxis')
title('Ice Cube Sample Clip')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
subplot(3,3,5)
spectrogram(dre(:,1), 'yaxis')
title('Dr. Dre Sample Clip')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
subplot(3,3,6)
spectrogram(dame(:,1), 'yaxis')
title('Dame D.O.L.L.A Sample Clip')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
subplot(3,3,7)
spectrogram(davis(:,1), 'yaxis')
title('Miles Davis Sample Clip')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
subplot(3,3,8)
spectrogram(coltrane(:,1), 'yaxis')
title('John Coltrane Sample Clip')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
subplot(3,3,9)
spectrogram(miller(:,1), 'yaxis')
title('Glen Miller Sample Clip')
xticks([0, ss * 1, ss * 2, ss * 3, ss * 4, ss * 5])
xticklabels({'0', '1', '2', '3', '4', '5'})
xlabel('Time (s)')
```

```
%% stack all three data matrices and get spectrogram for each clip
Dall = [mozart chopin beetho cube dame dre davis miller coltrane];
% classical: 1-100, rap: 101-200, jazz: 201-300
Sall = [];
for i=1:size(Dall,2)
    %sp = spectrogram(Dall(:, i), round(wlength));
    sp = spectrogram(Dall(:, i));
    Sall = [Sall, sp(:)];
end


%% svd
[u,s,v] = svd(Sall, 'econ');

%% plot singular value spectrum
figure
plot(diag(s)/sum(diag(s)), 'ro', 'Linewidth', 2)
title('Singular Value Spectrum')
ylabel('% Energy Captured')
xlabel('Singular Values')

%%
figure
scatter3(v(1:100,1), v(1:100,2), v(1:100,3), 'ko', 'Linewidth', 2)
hold on
%scatter3(v(101:200,1), v(101:200,2), v(101:200,3), 'ro', 'Linewidth', 2)
scatter3(v(201:300,1), v(201:300,2), v(201:300,3), 'bo', 'Linewidth', 2)
legend('classical', 'jazz') %'rap', 'jazz')
%%
% frommode = 1;
% tomode = 100;
% q1 = randperm(100);
% q2 = randperm(100);
% q3 = randperm(100);
%
% xice = v(1:100, frommode:tomode);
% xmd = v(101:200, frommode:tomode);
% xcho = v(201:300, frommode:tomode);
% xtrain = [xice(q1(1:80),:); xmd(q2(1:80),:); xcho(q3(1:80),:)];
% xtest = [xice(q1(81:end),:); xmd(q2(81:end),:); xcho(q3(81:end),:)];
% ytrain = zeros(240,1);
% for i=1:240
%     if i <= 80
%         ytrain(i) = 1;
%     elseif i <= 160
%             ytrain(i) = 2;
%     else
%         ytrain(i) = 3;
%     end
% end

[xtrain, ytrain, xtest, ytest] = train_test_split(v);
```

```
classifier = fitcnb(real(xtrain), ytrain);
predicted = predict(classifier, real(xtest));
figure
subplot(3,3,1)
bar(predicted(1:20))
axis([1 20 0 3])
title('predicting Classical (1 is correct)')
ylabel('test 1 predictions')
subplot(3,3,2)
bar(predicted(21:40))
axis([1 20 0 3])
title('predicting Rap (2 is correct)')
subplot(3,3,3)
bar(predicted(41:60))
axis([1 20 0 3])
title('predicting Jazz (3 is correct)')

[xtrain, ytrain, xtest, ytest] = train_test_split(v);
classifier = fitcknn(real(xtrain), ytrain);
predicted = predict(classifier, real(xtest));
subplot(3,3,4)
bar(predicted(1:20))
ylabel('test 2 predictions')
axis([1 20 0 3])
subplot(3,3,5)
bar(predicted(21:40))
axis([1 20 0 3])
subplot(3,3,6)
bar(predicted(41:60))
axis([1 20 0 3])

[xtrain, ytrain, xtest, ytest] = train_test_split(v);
%classifier = fitcknn(real(xtrain), ytrain, 'NumNeighbors', 5);
classifier = fitcnb(real(xtrain), ytrain);
predicted = predict(classifier, real(xtest));
subplot(3,3,7)
bar(predicted(1:20))
axis([1 20 0 3])
ylabel('test 3 predictions')
subplot(3,3,8)
bar(predicted(21:40))
axis([1 20 0 3])
subplot(3,3,9)
bar(predicted(41:60))
axis([1 20 0 3])
```

C Python Code

C.1 download_yt.py

```
# download youtube audios
from pytube import YouTube
from moviepy.editor import AudioFileClip
```

```
import pandas as pd


def download(url, fname):

    yt = YouTube(url)
    print('Downloading: ' + fname, str(yt.streams.filter(only_audio=True, subtype='mp4').
    yt.streams.filter(only_audio=True).first().download('data/', fname)

def main():
    song_list = pd.read_csv('song_list.csv')
    for idx, row in song_list.iterrows():
        if row['case'] != 5:
            continue
        download(row['url'], row['name'])


main()
```

C.2 mp42wav.py

```
# convert mp4 audios to .wav
from moviepy.editor import AudioFileClip
import glob

songs = glob.glob('data/*.mp4')
print(songs)
for song in songs:
    print(song)
    aclip = AudioFileClip(song)
    aclip.write_audiofile(song[:-3] + 'wav')
```