Amath482 – Winter 2020

Music Classification

Chenxi Di (1663044)

March 6, 2020

Abstract

This project aims to classify a given piece of music by sampling a 5 second clip. By using machine learning algorithms and SVD strategies, we could identify the genre/band of the music.

I. Introduction and Overview

Music genres are instantly recognizable to us, whether it be jazz, classical, blues. This project aims to develop a systematic way of classifying the genre and band of music by Matlab. By identifying certain features of the frequency and generating dominant spectrogram modes associated with a given band, we could more accurately classify the genre. Specifically, this project will attempt to classify three different bands of different genres (test 1), classify three different bands of same genre (test 2) and classify three genres from various bands.

II. Theoretical Background

The Singular Value Decomposition (SVD)

A singular value decomposition (SVD) is a factorization of a matrix into a some constitutive components all of which have useful meaning to the application. The full SVD decomposition of matrix A is:

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

where $\mathbf{U} \in C^{m\times m}$ is unitary, $\mathbf{V} \in C^{n\times n}$ is unitary and $\Sigma \in R^{m\times n}$ is diagonal.
To remove redundancy and identify the signals with maximal variance, we could diagonalize the covariance matrix, which is what the SVD does to reduce dimension. Note that the covariance matrix is

$$\mathbf{C_X} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^\mathbf{T} \tag{2}$$

where $\mathbf{X} \in C^{m\times n}$ where m are the measuring positions, and n is the number of data points at each position. The diagonal terms of $\mathbf{C_X}$ are the variances for measurements while the off-diagonal terms are the covariance. Large variances refers to dynamics of interest while low variances are considered to be non-interesting dynamics. By identifying the large variance, we could identify the most fluctuated points and therefore capture the important components.

Naive Bayes Classifier

In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong independence assumptions between the features. Using Bayes' theorem, the conditional probability can be decomposed as

$$P(C_k|x) = \frac{P(C_k)P(x|C_k)}{P(x)} \tag{3}$$

Based on this probability model, the Naive Bayes classifier becomes

$$\hat{y} = \underset{k\in(1\ldots K)}{\operatorname{argmax}} P(C_k)\Pi_{i=1}^{n} P(x_i|C_k) \tag{4}$$

Basically, the Naive Bayes algorithms will first compute the probability of the sample belonging to each categories. From here, it will assign the label to the sample with highest probability. As a supervised learning method, Naive Bayes algorithm usually performs well and fast despite its over simplified assumptions.

III. Algorithm Implementation and Development

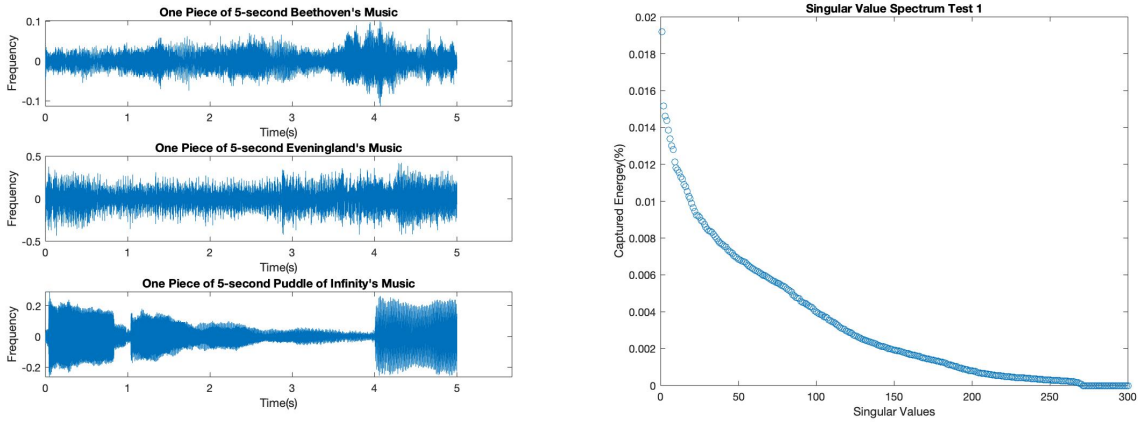
For the first test, I first loaded song files from Beethoven (classical | dark), Puddle of Infinity (ambient | calm), and Eveningland (dance and electronic). Based on the information for each audio files, I decided to randomly sample 100 5-seconds audio clips from each bands and labelled them. From here, I drew the time frequency graph for a sampled clip from each three bands and obtained the spectrograms by short time fourier transform.

With the spectrograms, we could therefore stack all three data and performed SVD to get a dominant spectrogram modes associated with a given band. When performing SVD, I used economy-size singular value decomposition to improve execution time and reduce the storage requirements without compromising the accuracy. By splitting the data into 80% training set and 20% testing set, we could train the Naive Bayes algorithm in the train set and predict the category in the test set, and acquired a prediction accuracy score. Other tests follow similar procedure.

IV. Computational Results

Test 1 Results

In test 1, I used 6 songs from three bands of completely different genres: Beethoven, classical and dark music; Eveningland, dance and electronic music; Puddle of Infinity, ambient and calm music. Obviously, from the time frequency plots, we can notice that these three music have different features. The Eveningland's music is characterized by its rhythmic and regular fluctuation, which is plausible since they are dance and electronic music. The Puddle of Infinity's music seems to be very calm at one period but fluctuate during another period, which also follows its ambient features. The Beethoven's music has some evident spikes.



(a) Time Frequency Plots

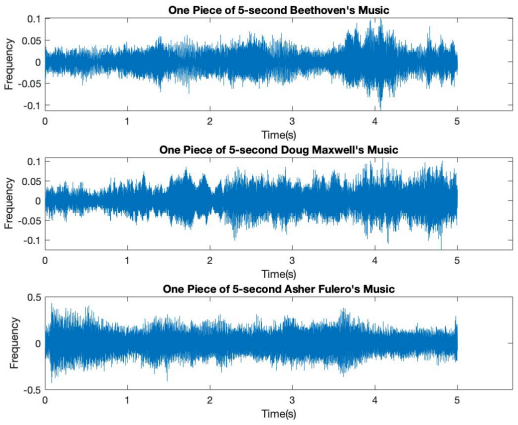
(b) Singular Value Spectrum from Three Bands

Figure 1: Test 1: 5-seconds Sample from Beethoven, Eveningland and Puddle of Infinity

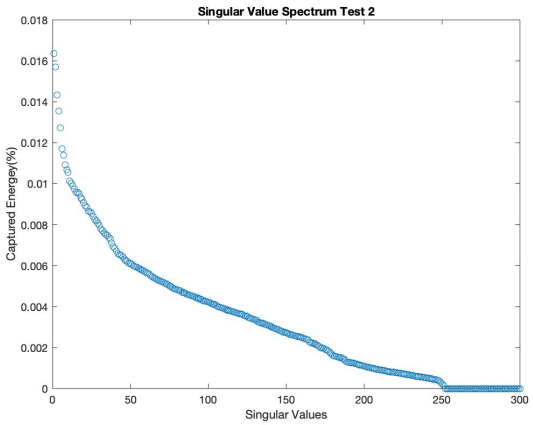
From the Singular Value Spectrum, only a few modes are more dominant. Most of the dominant modes capture around 1% energy. Using the first 100 modes in the training and test sets, the Naive Bayes classifier correctly categorizes 51 cases out of 60 cases in test sets. The classification accuracy achieves 85%, which is high. One possible reason for this is I selected three completely different bands, therefore the classifier is easier to capture the different features.

Test 2 Results

In test 2, I used 6 songs from 3 bands (Beethoven, Doug Maxwell, Asher Fulero) of the same genre - classical music. Different from first plot, the Figure2a shows that these song files do not have largely different frequency, probably because under the classical genre, most of the music are played by same musical instruments. However, the frequency of songs composed by Asher Fulero generally is much lower than other two sets of songs. When I listened to Asher's music, they are relatively steady and calm compared to other bands. From the Singular Value Spectrum, only a few modes are more dominant. Most of the dominant modes capture around 1% energy. Using the first 100 modes in the training and test sets, the Naive Bayes classifier correctly categorizes 43 cases out of 60 cases in test sets. The classification accuracy achieves 72%, which is lower than test1 since it is more difficult to detect band difference under the same genre.



(a) Time Frequency Plots

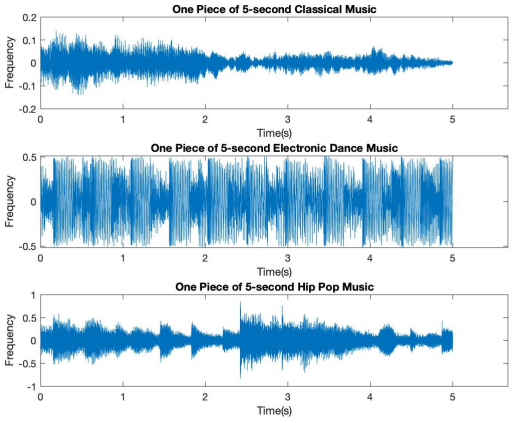


(b) Singular Value Spectrum from Three Bands

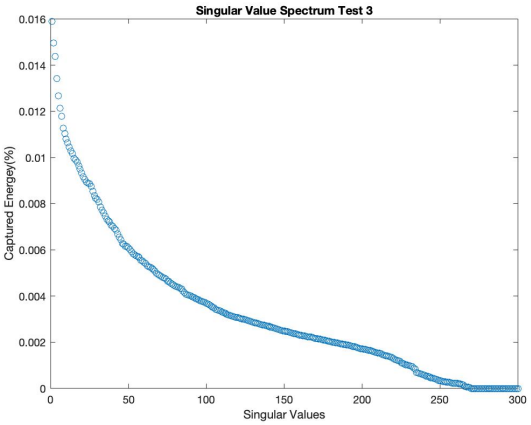
Figure 2: Test 2: 5-seconds Sample from Beethoven, Doug Maxwell, Asher Fulero

Test 3 Results

In test 3, I used 9 songs from different nine bands under three genres: classical music: Beethoven, Doug Maxwell, Asher Fulero; electronic and dance music: Geographer, Density and Time, Eveningland; Hip pop: The Grand Affair, Media Right Productions, Unicorn Heads. Figure 3 reveals that the electronic dance music is characterized by its rhythmic and regular fluctuation. On the other hand, Hip Pop music seems to have higher and more variant frequency compared to the classical music. From the Singular Value Spectrum, only a few modes are more dominant. Most of the dominant modes capture around 1% energy. Using the first 100 modes in the training and test sets, the Naive Bayes classifier also correctly categorizes 46 cases out of 60 cases in test sets. The classification accuracy achieves 76%, which is lower than test1 but higher than test 2. This is expected because even each bands have different styles, their genres will have noticeable differences, which makes the classifier easier to detect.



(a) Time Frequency Plots



(b) Singular Value Spectrum from Three Bands

Figure 3: Test 3: 5-seconds Sample from Classical , Electronic Dance and Hip Pop music

V. Summary and Conclusions

Over the course of the study, we extracted 100 5-seconds sample from each audio data in each test and performed SVD to obtain the dominant spectrogram modes. By Naive Bayes Classifier, we identify the bands and genres of each music clips in the test set. When classifying music clips from completely different bands, we achieved highest prediction accuracy. This happens since the clips' frequency varies a lot across different groups. With a larger amount of and more diverse music clips in the training set, the classifier is expected to achieve higher classification accuracy.

Appendix A MATLAB functions used and brief implementation explanation

`audioread` read audio files
`randperm(n, k)`: returns a row vector containing k unique integers selected randomly from 1 to n
`audioinfo`: get the information about the audio files
`spectrogram()`: returns the short-time Fourier transform of the input signal
`svd()`: perform a singular value decomposition of matrix X/sqrt(n-1)
`fitcnb`: fits a naive Bayes classifier
`predict`: predicts class label

Appendix B MATLAB codes

```
clear all; close all; clc;
%% band classification test 1

% get 100 random sampled 5-second pieces for Beethoven
info = audioinfo("Beethoven.wav"); % load 5th symphony, classical dark
time = info.Duration;
% get 50 random sampled 5-second pieces
Ab = [];  % store Beethoven signals
start_time = rand(1,50)*(time - 5)
```

```
% specify each period
start = round(start_time)*info.SampleRate;
for i = 1:50
    [y, Fs] = audioread("Beethoven.wav", [start(i), start(i) + 5*info.SampleRate]);
    Ab = [Ab sum(y,2)/size(y,2)];  % average two data sets for each channel
end
info = audioinfo("Moonlight.wav"); % load moonlight, classical sad
time = info.Duration;
start_time = rand(1,50)*(time - 5)
start = round(start_time)*info.SampleRate;
for i = 1:50
    [y, Fs] = audioread("Moonlight.wav", [start(i), start(i) + 5*info.SampleRate]);
    Ab = [Ab sum(y,2)/size(y,2)];  % average two data sets for each channel
end

% get 100 random sampled 5-second pieces for Eveningland
info = audioinfo("Nimbus.wav"); % load Nimbus, dance electronic
time = info.Duration;
Ae = [];
start_time = rand(1,50)*(time - 5)
% specify each period
start = round(start_time)*info.SampleRate;
for i = 1:50
    [y, Fs] = audioread("Nimbus.wav", [start(i), start(i) + 5*info.SampleRate]);
    Ae = [Ae sum(y,2)/size(y,2)];  % average two data sets for each channel
end

info = audioinfo("Nightingale.wav"); % load Nightingale, dance electronic
time = info.Duration;
start_time = rand(1,50)*(time - 5)
start = round(start_time)*info.SampleRate;
for i = 1:50
    [y, Fs] = audioread("Nightingale.wav", [start(i), start(i) + 5*info.SampleRate]);
    Ae = [Ae sum(y,2)/size(y,2)];  % average two data sets for each channel
end
```

```
% get 100 random sampled 5-second pieces for Puddle of Infinity
info = audioinfo("Wind.wav"); % load Wind Marching For Rain, ambient calm
time = info.Duration;
Ap = [];
start_time = rand(1,50)*(time - 5)
% specify each period
start = round(start_time)*info.SampleRate;
for i = 1:50
    [y, Fs] = audioread("Wind.wav", [start(i), start(i) + 5*info.SampleRate]);
    Ap = [Ap sum(y,2)/size(y,2)];  % average two data sets for each channel
end

info = audioinfo("Young.wav"); % load Young And Old Know Love, ambient calm
time = info.Duration;
start_time = rand(1,50)*(time - 5)
start = round(start_time)*info.SampleRate;
for i = 1:50
    [y, Fs] = audioread("Young.wav", [start(i), start(i) + 5*info.SampleRate]);
    Ap = [Ap sum(y,2)/size(y,2)];  % average two data sets for each channel
end



% plot signals
figure
subplot(3,1,1)
plot(Ab(:,1))
title("One Piece of 5-second Beethoven's Music")
xlabel("Time(s)")
ylabel("Frequency")
xticks([0, 220501/5, 220501*2/5, 220501*3/5, 220501*4/5, 220501]) % manually create t
xticklabels({'0', '1', '2', '3', '4', '5'})
subplot(3,1,2)
plot(Ae(:,1))
```

```
title("One Piece of 5-second Eveningland's Music")
xlabel("Time(s)")
ylabel("Frequency")
xticks([0, 220501/5, 220501*2/5, 220501*3/5, 220501*4/5, 220501]) % manually create tl
xticklabels({'0', '1', '2', '3', '4', '5'})
subplot(3,1,3)
plot(Ap(:,1))
title("One Piece of 5-second Puddle of Infinity's Music")
xlabel("Time(s)")
ylabel("Frequency")
xticks([0, 220501/5, 220501*2/5, 220501*3/5, 220501*4/5, 220501]) % manually create tl
xticklabels({'0', '1', '2', '3', '4', '5'})

% compute the spectrogram and perform SVD
A = [Ab, Ae, Ap];
S = [];
for i = 1:300
    sp = spectrogram(A(:,i));
    S = [S, sp(:)];
end
[u,s,v]=svd(S,'econ');

% plot singular
energy = diag(s)/sum(diag(s));
plot(energy,'o')
xlabel('Singular Values')
ylabel('Captured Energey(%)')
title('Singular Value Spectrum Test 1')

% use first 100 modes, split data into 8:2 training and testing, labelled the data
xb = v(1:100, 1:100);
xe = v(101:200, 1:100);
xp = v(201:300, 1:100);
index_helper1 = randperm(100);
index_helper2 = randperm(100);
```

```
index_helper3 = randperm(100);
xtrain = [xb(index_helper1(1:80),:); xe(index_helper2(1:80),:);xp(index_helper3(1:80)
ytrain = [zeros(80,1)+1;
          zeros(80,1)+2;
          zeros(80,1)+3];          % label y in training set

xtest = [xb(index_helper1(81:end),:); xe(index_helper2(81:end),:);xp(index_helper3(81
ytest = [zeros(20,1)+1;
          zeros(20,1)+2;
          zeros(20,1)+3];

% classification
classifier = fitcnb(real(xtrain),ytrain);
predict_value = predict(classifier,real(xtest));
% prediction accuracy
accuracy = sum(predict_value == ytest)/size(ytest,1);  % 85 percent



%% Test 2
% get 100 random sampled 5-second pieces for Doug Maxwell
info = audioinfo("Lost.wav"); % load lost in prayer, classical dark
time = info.Duration;
Ad = [];  % store Doug Maxwell signals
start_time = rand(1,50)*(time - 5)
% specify each period
start = round(start_time)*info.SampleRate;
for i = 1:50
    [y, Fs] = audioread("Lost.wav", [start(i), start(i) + 5*info.SampleRate]);
    Ad = [Ad sum(y,2)/size(y,2)];  % average two data sets for each channel
end
info = audioinfo("Bellissimo.wav"); % load Bellissimo, classical sad
time = info.Duration;
start_time = rand(1,50)*(time - 5)
start = round(start_time)*info.SampleRate;
for i = 1:50
```

```
    [y, Fs] = audioread("Bellissimo.wav", [start(i), start(i) + 5*info.SampleRate]);
    Ad = [Ad sum(y,2)/size(y,2)];  % average two data sets for each channel
end

% get 100 random sampled 5-second pieces for Asher Fulero
info = audioinfo("Aurora.wav"); % load Aurora Borealis Expedition, classical calm
time = info.Duration;
Aa = [];  % store Asher Fulero signals
start_time = rand(1,50)*(time - 5)
% specify each period
start = round(start_time)*info.SampleRate;
for i = 1:50
    [y, Fs] = audioread("Aurora.wav", [start(i), start(i) + 5*info.SampleRate]);
    Aa = [Aa sum(y,2)/size(y,2)];  % average two data sets for each channel
end
info = audioinfo("Confliction.wav"); % load Confliction & Catharsis, classical calm
time = info.Duration;
start_time = rand(1,50)*(time - 5)
start = round(start_time)*info.SampleRate;
for i = 1:50
    [y, Fs] = audioread("Confliction.wav", [start(i), start(i) + 5*info.SampleRate]);
    Aa = [Aa sum(y,2)/size(y,2)];  % average two data sets for each channel
end

% plot signals
figure
subplot(3,1,1)
plot(Ab(:,1))
title("One Piece of 5-second Beethoven's Music")
xlabel("Time(s)")
ylabel("Frequency")
xticks([0, 220501/5, 220501*2/5, 220501*3/5, 220501*4/5, 220501]) % manually create t
xticklabels({'0', '1', '2', '3', '4', '5'})
subplot(3,1,2)
plot(Ad(:,1))
```

```
title("One Piece of 5-second Doug Maxwell's Music")
xlabel("Time(s)")
ylabel("Frequency")
xticks([0, 220501/5, 220501*2/5, 220501*3/5, 220501*4/5, 220501]) % manually create tl
xticklabels({'0', '1', '2', '3', '4', '5'})
subplot(3,1,3)
plot(Aa(:,1))
title("One Piece of 5-second Asher Fulero's Music")
xlabel("Time(s)")
ylabel("Frequency")
xticks([0, 220501/5, 220501*2/5, 220501*3/5, 220501*4/5, 220501]) % manually create tl
xticklabels({'0', '1', '2', '3', '4', '5'})

% compute the spectrogram and perform SVD
A = [Ab, Ad, Aa];
S = [];
for i = 1:300
    sp = spectrogram(A(:,i));
    S = [S, sp(:)];
end
[u,s,v]=svd(S,'econ');

% plot singular
energy = diag(s)/sum(diag(s));
plot(energy,'o')
xlabel('Singular Values')
ylabel('Captured Energey(%)')
title('Singular Value Spectrum Test 2')

% use first 100 modes, split data into 8:2 training and testing, labelled the data
xb = v(1:100, 1:100);
xd = v(101:200, 1:100);
xa = v(201:300, 1:100);
index_helper1 = randperm(100);
index_helper2 = randperm(100);
```

```
index_helper3 = randperm(100);
xtrain = [xb(index_helper1(1:80),:); xd(index_helper2(1:80),:);xa(index_helper3(1:80)
ytrain = [zeros(80,1)+1;
          zeros(80,1)+2;
          zeros(80,1)+3];          % label y in training set

xtest = [xb(index_helper1(81:end),:); xd(index_helper2(81:end),:);xa(index_helper3(81
ytest = [zeros(20,1)+1;
          zeros(20,1)+2;
          zeros(20,1)+3];

% classification
classifier = fitcnb(real(xtrain),ytrain);
predict_value = predict(classifier,real(xtest));
% prediction accuracy
accuracy = sum(predict_value == ytest)/size(ytest,1);  % 72 percent

%% test 3
% get 100 random sampled 5-second pieces for classical music, three bands
info = audioinfo("Lost.wav"); % load lost in prayer, Doug Maxwell, dark
time = info.Duration;
Ac = [];  % store classical music signals
start_time = rand(1,40)*(time - 5)
% specify each period
start = round(start_time)*info.SampleRate;
for i = 1:40
    [y, Fs] = audioread("Lost.wav", [start(i), start(i) + 5*info.SampleRate]);
    Ac = [Ac sum(y,2)/size(y,2)];  % average two data sets for each channel
end
info = audioinfo("Confliction.wav"); % load Confliction & Catharsis, Asher Fulero, ca
time = info.Duration;
start_time = rand(1,30)*(time - 5)
start = round(start_time)*info.SampleRate;
for i = 1:30
    [y, Fs] = audioread("Confliction.wav", [start(i), start(i) + 5*info.SampleRate]);
```

```
    Ac = [Ac sum(y,2)/size(y,2)];  % average two data sets for each channel
end
info = audioinfo("Moonlight.wav"); % load moonlight, Beethoven, classical sad
time = info.Duration;
start_time = rand(1,30)*(time - 5)
start = round(start_time)*info.SampleRate;
for i = 1:30
    [y, Fs] = audioread("Moonlight.wav", [start(i), start(i) + 5*info.SampleRate]);
    Ac = [Ac sum(y,2)/size(y,2)];  % average two data sets for each channel
end

% get 100 random sampled 5-second pieces for edm, three bands
info = audioinfo("Nimbus.wav"); % load Nimbus, dance electronic
time = info.Duration;
Aedm = [];
start_time = rand(1,50)*(time - 5)
% specify each period
start = round(start_time)*info.SampleRate;
for i = 1:50
    [y, Fs] = audioread("Nimbus.wav", [start(i), start(i) + 5*info.SampleRate]);
    Aedm = [Aedm sum(y,2)/size(y,2)];  % average two data sets for each channel
end
info = audioinfo("Sky.wav"); % load Skycrapper, Geographer
time = info.Duration;
start_time = rand(1,30)*(time - 5)
start = round(start_time)*info.SampleRate;
for i = 1:30
    [y, Fs] = audioread("Sky.wav", [start(i), start(i) + 5*info.SampleRate]);
    Aedm = [Aedm sum(y,2)/size(y,2)];  % average two data sets for each channel
end
info = audioinfo("Venetian.wav"); % by Density & Time
time = info.Duration;
start_time = rand(1,30)*(time - 5)
start = round(start_time)*info.SampleRate;
for i = 1:30
```

```
    [y, Fs] = audioread("Venetian.wav", [start(i), start(i) + 5*info.SampleRate]);
    Aedm = [Aedm sum(y,2)/size(y,2)];  % average two data sets for each channel
end

% get 100 random sampled 5-second pieces for Hip pop, three bands
info = audioinfo("Orange.wav");
time = info.Duration;
Ah = [];
start_time = rand(1,50)*(time - 5)
% specify each period
start = round(start_time)*info.SampleRate;
for i = 1:50
    [y, Fs] = audioread("Orange.wav", [start(i), start(i) + 5*info.SampleRate]);
    Ah = [Ah sum(y,2)/size(y,2)];  % average two data sets for each channel
end
info = audioinfo("Emotional.wav");
time = info.Duration;
start_time = rand(1,30)*(time - 5)
start = round(start_time)*info.SampleRate;
for i = 1:30
    [y, Fs] = audioread("Emotional.wav", [start(i), start(i) + 5*info.SampleRate]);
    Ah = [Ah sum(y,2)/size(y,2)];  % average two data sets for each channel
end
info = audioinfo("Jane.wav");
time = info.Duration;
start_time = rand(1,30)*(time - 5)
start = round(start_time)*info.SampleRate;
for i = 1:30
    [y, Fs] = audioread("Jane.wav", [start(i), start(i) + 5*info.SampleRate]);
    Ah = [Ah sum(y,2)/size(y,2)];  % average two data sets for each channel
end

% plot signals
figure
subplot(3,1,1)
```

```
plot(Ac(:,1))
title("One Piece of 5-second Classical Music")
xlabel("Time(s)")
ylabel("Frequency")
xticks([0, 220501/5, 220501*2/5, 220501*3/5, 220501*4/5, 220501]) % manually create tl
xticklabels({'0', '1', '2', '3', '4', '5'})
subplot(3,1,2)
plot(Aedm(:,1))
title("One Piece of 5-second Electronic Dance Music")
xlabel("Time(s)")
ylabel("Frequency")
xticks([0, 220501/5, 220501*2/5, 220501*3/5, 220501*4/5, 220501]) % manually create tl
xticklabels({'0', '1', '2', '3', '4', '5'})
subplot(3,1,3)
plot(Ah(:,1))
title("One Piece of 5-second Hip Pop Music")
xlabel("Time(s)")
ylabel("Frequency")
xticks([0, 220501/5, 220501*2/5, 220501*3/5, 220501*4/5, 220501]) % manually create tl
xticklabels({'0', '1', '2', '3', '4', '5'})

% compute the spectrogram and perform SVD
A = [Ac, Aedm, Ah];
S = [];
for i = 1:300
    sp = spectrogram(A(:,i));
    S = [S, sp(:)];
end
[u,s,v]=svd(S,'econ');

% plot singular
energy = diag(s)/sum(diag(s));
plot(energy,'o')
xlabel('Singular Values')
ylabel('Captured Energey(%)')
```

```
title('Singular Value Spectrum Test 3')

% use first 100 modes, split data into 8:2 training and testing, labelled the data
xc = v(1:100, 1:100);
xedm = v(101:200, 1:100);
xh = v(201:300, 1:100);
index_helper1 = randperm(100);
index_helper2 = randperm(100);
index_helper3 = randperm(100);
xtrain = [xc(index_helper1(1:80),:); xedm(index_helper2(1:80),:);xh(index_helper3(1:8
ytrain = [zeros(80,1)+1;
          zeros(80,1)+2;
          zeros(80,1)+3];          % label y in training set

xtest = [xc(index_helper1(81:end),:); xedm(index_helper2(81:end),:);xh(index_helper3(
ytest = [zeros(20,1)+1;
          zeros(20,1)+2;
          zeros(20,1)+3];

% classification
classifier = fitcnb(real(xtrain),ytrain);
predict_value = predict(classifier,real(xtest));
% prediction accuracy
accuracy = sum(predict_value == ytest)/size(ytest,1);  % 72 percent
```

Reference

MathWorks: https://www.mathworks.com/help/matlab/ref/

J. Nathan Kutz. (2013) Data-Driven Modeling and Scientific Computation: Methods for Complex Systems and Big Data. Oxford University Press. ISBN:978-0-19-966034-6