

Homework 3: Music Genre Identification

DUE: Thursday, Feb. 22, 2018

Music genres are instantly recognizable to us, whether it be jazz, classical, blues, rap, rock, etc. One can always ask how the brain classifies such information and how it makes a decision based upon hearing a new piece of music. The objective of this homework is to attempt to write a code that can classify a given piece of music by sampling a 5 second clip.

As an example, consider Fig. 1. Four classic pieces of music are demonstrated spanning genres of rap, jazz, classic rock and classical. Specifically, a 3-second sample is given of Dr. Dre's *Nuthin' but a 'G' thang* (The Chronic), John Coltrane's *A Love Supreme* (A Love Supreme), Led Zeppelin's *Over The Hills and Far Away* (Houses of the Holy), and Mozart's *Kyrie* (Requiem). Each has a different signature, thus begging the question whether a computer could distinguish between genres based upon such a characterization of the music.

- **(test 1) Band Classification:** Consider three different bands of your choosing and of different genres. For instance, one could pick Michael Jackson, Soundgarden, and Beethoven. By taking 5-second clips from a variety of each of their music, i.e. building training sets, see if you can build a statistical testing algorithm capable of accurately identifying "new" 5-second clips of music from the three chosen bands.
- **(test 2) The Case for Seattle:** Repeat the above experiment, but with three bands from within the same genre. This makes the testing and separation much more challenging. For instance, one could focus on the late 90s Seattle grunge bands: Soundgarden, Alice in Chains, and Pearl Jam. What is your accuracy in correctly classifying a 5-second sound clip? Compare this with the first experiment with bands of different genres.
- **(test 3) Genre Classification:** One could also use the above algorithms to simplify broadly classify songs as jazz, rock, classical etc. In this case, the training sets should be various bands within each genre. For instance, classic rock bands could be classified using sounds clips from Zep, AC/DC, Floyd, etc. while classical could be classified using Mozart, Beethoven, Bach, etc. Perhaps you can limit your results to three genres, for instance, rock, jazz, classical.

WARNING and NOTES: You will probably want to SVD the spectrogram of songs versus the songs themselves. Interestingly, this will give you the dominant spectrogram *modes* associated with a given band. Moreover, you may want to re-sample your data (i.e. take every other point) in order to keep the data sizes more manageable. Regardless, you will need lots of processing time.

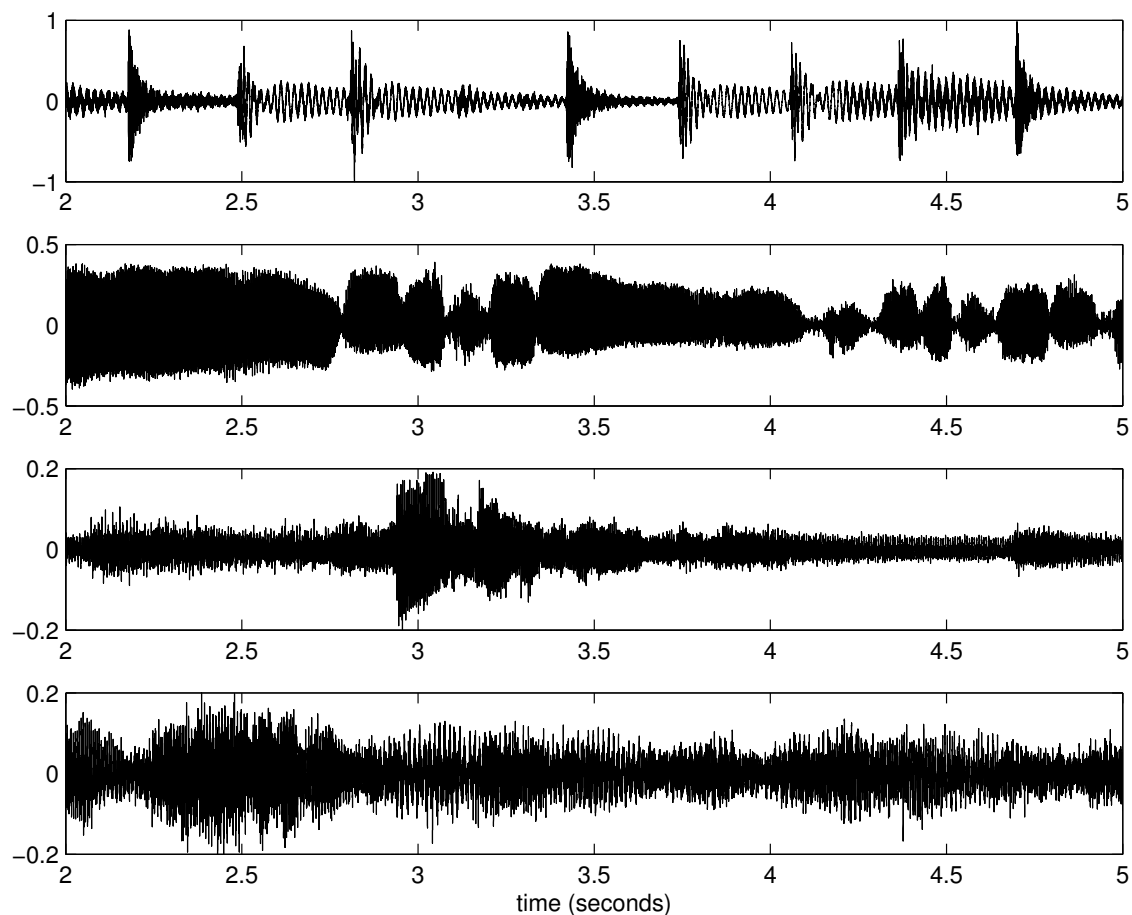


Figure 1: Instantly recognizable, these four pieces of music are (in order of top to bottom): Dr. Dre's *Nuthin' but a 'G' thang* (The Chronic), John Coltrane's *A Love Supreme* (A Love Supreme), Led Zeppelin's *Over The Hills and Far Away* (Houses of the Holy), and Mozart's *Kyrie* (Requiem). Illustrated is a 3-second clip from time 2 seconds to 5 seconds of each of these songs.

AMATH 582 Computational Methods for Data Analysis

Music Genre Classification

By,

Ruchit Patel

Master of Science in Mechanical Engineering

University of Washington, Seattle

Abstract

This report contains genre classification using supervised machine learning algorithm famously known as Naive Bayes as it can classify between more than two groups. Three different tests are performed which are; band classification from three different bands having different genres, band classification from three different bands having the same genre, genre classification from three different genres. To do this, SVD on the real part of spectrogram version of randomly chosen five seconds clip from 100 songs per genre is vectorized to make training data set. Then 80%-20% cross-validation is done to check training accuracy and then outside data is fed to Matlab to check its testing accuracy. Many iterations are done by changing few parameters every time and results of those are shown in section five. All the training sets are showing good, ~80-90%, accuracy in cross-validation and the testing accuracy of ~70% for the test-1, ~50% for the test-2 and ~70% for the test-3 is obtained.

Section [I] Introduction and Overview

On an average, 75,000 music albums are launched every year. Classifying each of the songs to that genre has already become a gigantic and difficult task. So, there should be some mechanism which can do this. Our brain can predict the genre of the song just by listening to it. So, there must be some way to teach a computer to predict the genre. This is where machine learning classification of genres comes in to picture and the benefit is that there are already millions of songs are classified which can be used to train the machine by using supervised classification.

With time, people have tried different things to get robust features which can predict genre accurately. They have tried using direct signals of audio as data matrix, people have tried to find pattern between maximum frequency, mean and variance of songs of different genres, some people have also tried to find pattern in change in frequency by computing derivatives at each point, another way of doing is FFT or DCT of audio clip to find pattern between frequency modes, but out of all this, doing spectrogram of audio clips gives the best classified training set.

Here, 100 songs from each genre, classical-pop-rock are taken and then a sample of five seconds from each of the songs are chosen randomly to give robustness to the dataset. In conclusion, it is shown that samples of five seconds are enough to predict the genre and this gives almost similar accuracy as a sample having 30 seconds of an audio clip which is surprising. Training accuracy between datasets made using real, imaginary, both and absolute values of the spectrogram is also compared and results are shown in section IV.

To find training accuracy, the average accuracy of 100 iterations is considered and in every iteration, Matlab takes 80 samples as training set and 20 samples as test data randomly from \mathbf{V} matrix obtained from SVD of the data matrix. This 100 times cross-validation is done using Gaussian Mixture Models, Naive Bayes, Linear Discriminant Analysis, Simple Vector Machine, and Regression Tree. Out of this, by making some compromises between accuracy, robustness, and difficulty in implementation, Naive Bayes is chosen to do further analysis.

At last finally, extra songs, which are not part of the trained dataset, are given to machine and the test accuracy is measured using Naive Bayes algorithm and conclusions from it for different tests are discussed in section V. Here, comparison of test accuracy while using different number of features is done to find peak accuracy and significant variation in accuracy is observed with respect number of feature used.

Section [II] Theoretical Background

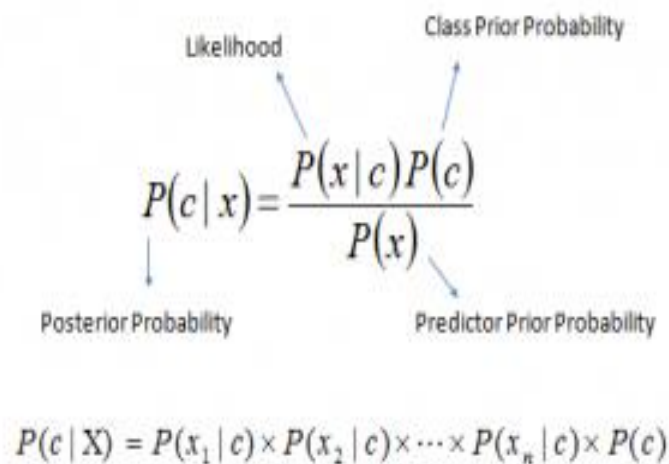
As we know, matrix multiplied with vector gives another vector. We also know that any vector can be represented by its direction and magnitude in that direction. This tells that all matrix multiplication do is to stretch/compress and rotate the vector. Also, Stretching and rotating can be described independently. This is the baseline of Singular Value Decomposition.

Say, there is an n -dimensional unit sphere in orthogonal n -dimensional vector space of $[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \dots, \mathbf{v}_n]$, now by multiplying it with any matrix \mathbf{A} , the original sphere has now been rotated and stretched/ compressed into the n -dimensional ellipse. If the resultant ellipse is in orthogonal n -dimensional space of $[\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \dots, \mathbf{u}_n]$ and if magnitude in all U directions is $\sigma_1, \sigma_2, \dots, \sigma_n$ respectively then we can write that $\mathbf{AV} = \mathbf{U}\Sigma$. So now $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$. Here, \mathbf{U} and \mathbf{V} are unitary matrices and their inverse is equal to their transpose and Σ is a diagonal matrix.

This process of decomposing a matrix in a mentioned way is called Singular Value Decomposition. This decomposition is possible for every single matrix and algorithm work in such a way that singular values of Σ come in decreasing order according to their value. Here, first singular values and corresponding first singular mode contains the highest amount of information of whole dataset and amount of information contain decreases with the further singular modes.

The second major mathematical function used here is one of the supervised machine learning classifier called Naive Bayes Classifier. In machine learning, naive Bayes classifiers are a family of

simple probabilistic classifiers based on applying Bayes' theorem with independence assumptions between the features.



$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Posterior Probability

Likelihood

Class Prior Probability

Predictor Prior Probability

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

Figure (1) Naive Bayes Probability Equation^[3]

In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.

Section [III] Algorithm Implementation

First of all, after downloading the database, all audio clips are imported and read into Matlab using 'dir' and 'audioread' command respectively. Then every song is vectorized and stores into data matrix column-by-column after converting its randomly chosen five-second clip into a spectrogram.

Second, now SVD is performed on this data matrix and from this **V** matrix containing features is used for further analysis. This matrix is distributed into train-set and test-set in 80%-20% proportion randomly to do cross-validation by using Naïve Bayes algorithm. Cross-validation is done 100 times and average training accuracy is determined.

Third, to determine test accuracy, which is the main objective, 10 songs, which are not part of the dataset on which SVD is performed, are given as test set and hundreds of iterations are done by changing a number of features used every time to determine the maximum possible accuracy.

This mentioned process is repeated for each of the case. The only difference is in the input data set and desired output.

Section [IV] Computational Results

Test-1:

For the band classification task, three artists having different genres are chosen which are Alizee (Pop-Japan), Eminem (Rap-USA) and Kailasa (Classic-India). Above mentioned algorithm is followed and training and testing accuracies are determined. As these are artists from very different genres, high classifier accuracy was expected.

At first, accuracy was poor, somewhere around 50%, so then instead of using 100 songs per artist, which I have done for other tests, I increased the training dataset up to 160 songs per artist. Also, after realizing a number of features used from **V** matrix can affect the accuracy, 'for' loops are made which goes through every possible sequential combination from 1 to 480 and determine the accuracy at that time. The code ran for total 4 hours and optimum range is found out.

The optimum range for testing accuracy for this database when Naive Bayes is used is 87-240. Using these exact columns of **V** matrix is giving 70%+ accuracy. Of course, these would change if any of the other parameter is changed in the algorithm.

Test-2:

Here, I took late 90s Seattle grudge bands as my dataset which are Soundgarden, Alice in Chains and Pearl Jam. As these bands are within the same genre, low classifier accuracy was predicted. Again exactly same algorithm was implemented and spectrogram of 5 seconds sound clip was used as training dataset.

Here, after changing many numbers of parameters the highest 50% test-accuracy was obtained, The parameter for this accuracy were these: Imaginary numbers of spectrogram as a dataset, mean of both column of stereo sound, 7-49 columns of **V** matrix.

Test-3:

For genre classification, I had the 100 songs of 10 different genres as a total dataset. First, I started comparing two genres with each other to determine which genres are easy to classify and which are difficult. After comparing a bunch of two different genres; classical, pop and rock, all were giving 80%+ training accuracy when used with each other. So, finally, I decided to make a combination of these three to make my training set.

So, for test-3, my dataset is consist of songs from classical, pop and rock. 100 sound clips are used for each song. After 100 cross-validations, 76.33% average training accuracy was obtained.

Section [V] Summary and Conclusion

1) On doing cross-validation using GMM, NB, LDA, SVM, and CART; I can conclude that there is not much accuracy difference ($\pm 5\%$) between classifier models but Naive Bayes seems easiest to

implement as there is no constrain regarding how much number of the features are used. GMM, SVM, and LDA are sensitive regarding a number of features used. Also, SVM doesn't allow more than 2 groups to be classified.

2) After choosing Naive Bayes as my model, cross-validation of the dataset having [5, 10, 15, 20, 25] seconds of the audio clip is found. Here, I have concluded that five seconds of the audio clip is totally enough to predict the genre as training accuracy remained almost constant ($\pm 2\%$) with an increase in the size of the audio clip.

3) Now, after finalizing five second as sufficient data, turn-by-turn real part, imaginary part, both parts and absolute value of spectrogram is used to make data matrix and again it is found that there is no much difference between accuracy ($\pm 3\%$) and so I have used absolute part of spectrogram for my further analysis which takes care of real as well as imaginary part.

4) For audio files having a stereo data type, four iterations are done. First only left column, second only right column, third both columns and the fourth mean of both columns. It is found that mean of both columns gives the little bit higher accuracy than other results hence it is used for all other datasets.

5) After optimizing mentioned parameters concerning training accuracy, now focus was transferred on test accuracy. Here, there is only one new parameter which can vary the accuracy and that is a number of features used to train the classifier model. Here, test accuracy using every single feature alone as well as every feature with the combination of all other features are evaluated by making big 'for' loop doing thousands of iterations. Drastic variation in the accuracy is observed and a group of features to be used to get maximum accuracy is found and obviously, it is different for all test.

6) In test-1 containing different artist/band from a different genre, 90%+ cross-validation accuracy and ~70%+ test accuracy is found which are very satisfying and also expected results, as data set, has a lot of dissimilarities.

7) As test-2 contains bands from the same genre, ~70% training accuracy, and ~50% test accuracy is obtained which are very less compared to other tests mostly because as similarities increases, it becomes harder and harder to find distinguishing feature between them. Here, as a future work, very big dataset and some modern machine learning algorithm such as deep neural network can be implemented to increase the accuracy.

8) Test-3 contains songs from a different genre and here again, ~90%+ training accuracy and ~70%+ test accuracy is observed. Good thing to notice here was that whenever classical songs are used with any other songs, accuracy goes higher as classical is very different genre than most of the other genre. But, when blue and country or country and jazz are used, the accuracy goes drastically down as they are almost similar types of music genres.

References

- 1) Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data, J. Nathan Kutz, ISBN: 978-0199660346
- 2) <https://www.mathworks.com/help/matlab>
- 3) <https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/>
- 4) https://en.wikipedia.org/wiki/Naive_Bayes_classifier
- 5) Audio-file for the Dataset from:
<https://freemusicdownloads.world/>
http://marsyasweb.appspot.com/download/data_sets/

Acknowledgment

I am thankful to Prof. Jose Nathan Kutz for their interesting lectures and providing an opportunity to do machine learning in this homework and I am very grateful to TAs, Stritam and Jize for their support in all parts of the homework.

Appendix A

dir – list folder contents

audioread – reads audio file

strcat – concatenate strings horizontally

svd – computes the Singular Value Decomposition of an array

randperm – random permutation of numbers

fitgmdist – fits Gaussian mixture distribution to the data

fitcnb – train multi class Naive Bayes Model

svmtrain – trains Support Vector Machine classifier

Appendix B

```
%% Importing Audio
path_to_classical = 'U:\AMATH 582 - Computational Method for Data
Analysis\HW03\Genres\genres\classical\';
classical_file = dir(path_to_classical);
path_to_pop = 'U:\AMATH 582 - Computational Method for Data
Analysis\HW03\Genres\genres\pop\';
pop_file = dir(path_to_pop);
path_to_rock = 'U:\AMATH 582 - Computational Method for Data
Analysis\HW03\Genres\genres\rock\';
rock_file = dir(path_to_rock);
%%
S_classical = [];
for ii=3:length(classical_file)
[a,fps] = audioread(strcat(path_to_classical,classical_file(ii).name));
rand = randperm(size(a,1)-5*fps,1);
audio_classical(:,ii-2) = a(rand:rand+5*fps,1);
spectro_classical = real(spectrogram(audio_classical(:,ii-2)));
S_classical = [S_classical spectro_classical(:)];
clear a spectro_classical;
end
%%
% fourier_blue = real(fft(audio_blue));
% fourier_blue = real(spectrogram(audio_blue));
%%
S_pop = [];
for ii=3:length(pop_file)
[a,fps] = audioread(strcat(path_to_pop,pop_file(ii).name));
rand = randperm(size(a,1)-5*fps,1);
audio_pop(:,ii-2) = a(rand:rand+5*fps,1);
spectro_pop = real(spectrogram(audio_pop(:,ii-2)));
S_pop = [S_pop spectro_pop(:)];
clear a spectro_pop;
end
%%
% fourier_pop = real(fft(audio_pop));
% fourier_pop = real(spectrogram(audio_pop));
%%
S_rock = [];
for ii=3:length(rock_file)
[a,fps] = audioread(strcat(path_to_rock,rock_file(ii).name));
rand = randperm(size(a,1)-5*fps,1);
audio_rock(:,ii-2) = a(rand:rand+5*fps,1);
spectro_rock = real(spectrogram(audio_rock(:,ii-2)));
S_rock = [S_rock spectro_rock(:)];
clear a spectro_rock;
end
%%
% data = [fourier_blue fourier_pop];
% data = [audio_blue audio_pop];
data = [S_classical(:,1:90) S_pop(:,1:90) S_rock(:,1:90)];
%%
% [m,n] = size(data);
% mn = mean(data,2);
```

```

% data = data - repmat(mn,1,n);
[u,s,v] = svd(data,'econ');
%%
v_classical = v(1:90,:);
v_pop = v(91:180,:);
v_rock = v(181:end,:);
v_test_classical = (inv(s)*u'*S_classical(:,91:end))';
v_test_pop = (inv(s)*u'*S_pop(:,91:100))';
v_test_rock = (inv(s)*u'*S_rock(:,91:100))';
%%
% for l=1:100
for m=1:90
for n=m:90
% q1 = randperm(100); q2 = randperm(100); q3 = randperm(100);
% train = [v_classical(q1(1:80),:); v_pop(q2(1:80),:); v_rock(q3(1:80),:)]';
train = [v_classical(:,m:n); v_pop(:,m:n); v_rock(:,m:n)];
% test = [v_classical(q1(81:end),:); v_pop(q2(81:end),:);
v_rock(q3(81:end),:)]';
test = [v_test_classical(:,m:n); v_test_pop(:,m:n); v_test_rock(:,m:n)];
% label = [ones(80,1); 2*ones(80,1); 3*ones(80,1)];
label = [ones(90,1); 2*ones(90,1); 3*ones(90,1)];

% gm = fitgmdist(train,2,'CovarianceType','diagonal');
% pre = cluster(gm,test);
% figure; bar(pre)

nb = fitcnb(train,label);
pre = nb.predict(test);
% bar(pre)

% svm = svmtrain(train,label);
% pre = svmclassify(svm,test);
% figure; bar(pre);

actual = [ones(10,1); 2*ones(10,1); 3*ones(10,1)];
test_accuracy(n,m) = (sum((actual-pre)==0)/30)*100;

% actual = [ones(20,1); 2*ones(20,1); 3*ones(20,1)];
% train_accuracy(l,1) = (sum((actual-pre)==0)/60)*100;

end
end
% end
% mean(train_accuracy)

clear all;
%% Importing Audio
path_to_alice = 'U:\AMATH 582 - Computational Method for Data
Analysis\HW03\Test-2\alice\';
alice_file = dir(path_to_alice);
path_to_pearl = 'U:\AMATH 582 - Computational Method for Data
Analysis\HW03\Test-2\pearl\';
pearl_file = dir(path_to_pearl);
path_to_soundgarden = 'U:\AMATH 582 - Computational Method for Data
Analysis\HW03\Test-2\soundgarden\';

```

```

soundgarden_file = dir(path_to_soundgarden);
%%
S_alice = [];
for ii=3:length(alice_file)
[a,fps] = audioread(strcat(path_to_alice,alice_file(ii).name));
% a = a(:);
a = mean(a,2);
for jj=16:1:31
audio_alice = a(jj*5*fps+1:(jj+1)*5*fps);
spectro_alice = imag(spectrogram(audio_alice));
S_alice = [S_alice spectro_alice(:)];
clear spectro_alice audio_alice;
end
clear a;
end
%%
% fourier_blue = real(fft(audio_blue));
% fourier_blue = real(spectrogram(audio_blue));
%%
S_pearl = [];
for ii=3:length(pearl_file)
[a,fps] = audioread(strcat(path_to_pearl,pearl_file(ii).name));
% a = a(:);
a = mean(a,2);
for jj=16:1:31
audio_pearl = a(jj*5*fps+1:(jj+1)*5*fps);
spectro_pearl = imag(spectrogram(audio_pearl));
S_pearl = [S_pearl spectro_pearl(:)];
clear audio_pearl spectro_pearl;
end
clear a;
end
%%
% fourier_pearl = real(fft(audio_pearl));
% fourier_pearl = real(spectrogram(audio_pearl));
%%
S_soundgarden = [];
for ii=3:length(soundgarden_file)
[a,fps] = audioread(strcat(path_to_soundgarden,soundgarden_file(ii).name));
% a = a(:);
a = mean(a,2);
for jj=16:1:31
audio_soundgarden = a(jj*5*fps+1:(jj+1)*5*fps);
spectro_soundgarden = imag(spectrogram(audio_soundgarden));
S_soundgarden = [S_soundgarden spectro_soundgarden(:)];
clear audio_soundgarden spectro_soundgarden;
end
clear a;
end
%%
% data = [fourier_blue fourier_pearl];
% data = [audio_blue audio_pearl];
data = [S_alice(:,1:160) S_pearl(:,1:160) S_soundgarden(:,1:160)];

% [m,n] = size(data);
% mn = mean(data,2);
% data = data - repmat(mn,1,n);

```

```

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

v_alice = v(1:160,:);
v_pearl = v(161:320,:);
v_soundgarden = v(321:end,:);
% v_test_blue = (inv(s)*u'*S_blue(:,91:end))';
% v_test_pearl = (inv(s)*u'*S_pearl(:,91:100))';
%%
% for l=1:100
%     m = 1; n = 480;
for m=1:480
for n=m:480
q1 = randperm(160); q2 = randperm(160); q3 = randperm(160);
train = [v_alice(q1(1:145),m:n); v_pearl(q2(1:145),m:n);
v_soundgarden(q3(1:145),m:n)];
% train = [v_alice(:,m:n); v_pearl(:,m:n)];
test = [v_alice(q1(146:end),m:n); v_pearl(q2(146:end),m:n);
v_soundgarden(q3(146:end),m:n)];
% test = [v_test_alice(:,m:n); v_test_pearl(:,m:n)];
label = [ones(145,1); 2*ones(145,1); 3*ones(145,1)];
% label = [ones(90,1); 2*ones(90,1)];

% gm = fitgmdist(train,2,'CovarianceType','diagonal');
% pre = cluster(gm,test);
% figure; bar(pre)

nb = fitcnb(train,label);
pre = nb.predict(test);
% bar(pre)

% pre = classify(test,train,label,'diagquadratic');
% bar(pre)

% svm = svmtrain(train,label);
% pre = svmclassify(svm,test);
% figure; bar(pre);

% actual = [ones(10,1); 2*ones(10,1)];
% test_accuracy(n,m) = (sum((actual-pre)==0)/20)*100;

actual = [ones(15,1); 2*ones(15,1); 3*ones(15,1)];
train_accuracy(n,m) = (sum((actual-pre)==0)/45)*100;

end
end
% end
mean(train_accuracy)
% 21-174 91%
% max(test_accuracy)

clear all;
%% Importing Audio
path_to_alizee = 'U:\AMATH 582 - Computational Method for Data
Analysis\HW03\Test-3\alizee\';
alizee_file = dir(path_to_alizee);

```

```

path_to_eminem = 'U:\AMATH 582 - Computational Method for Data
Analysis\HW03\Test-3\eminem\';
eminem_file = dir(path_to_eminem);
path_to_kailasa = 'U:\AMATH 582 - Computational Method for Data
Analysis\HW03\Test-3\kailasa\';
kailasa_file = dir(path_to_kailasa);
%%
S_alizee = [];
for ii=3:length(alizee_file)
[a,fps] = audioread(strcat(path_to_alizee,alizee_file(ii).name));
% a = a(:);
a = mean(a,2);
for jj=8:1:29
audio_alizee = a(jj*5*fps+1:(jj+1)*5*fps);
spectro_alizee = abs(spectrogram(audio_alizee));
S_alizee = [S_alizee spectro_alizee(:)];
clear spectro_alizee audio_alizee;
end
clear a;
end
%%
% fourier_blue = real(fft(audio_blue));
% fourier_blue = real(spectrogram(audio_blue));
%%
S_eminem = [];
for ii=4:length(eminem_file)
[a,fps] = audioread(strcat(path_to_eminem,eminem_file(ii).name));
% a = a(:);
a = mean(a,2);
for jj=8:1:29
audio_eminem = a(jj*5*fps+1:(jj+1)*5*fps);
spectro_eminem = abs(spectrogram(audio_eminem));
S_eminem = [S_eminem spectro_eminem(:)];
clear audio_eminem spectro_eminem;
end
clear a;
end
%%
% fourier_eminem = real(fft(audio_eminem));
% fourier_eminem = real(spectrogram(audio_eminem));
%%
S_kailasa = [];
for ii=3:length(kailasa_file)
[a,fps] = audioread(strcat(path_to_kailasa,kailasa_file(ii).name));
% a = a(:);
a = mean(a,2);
for jj=8:1:29
audio_kailasa = a(jj*5*fps+1:(jj+1)*5*fps);
spectro_kailasa = abs(spectrogram(audio_kailasa));
S_kailasa = [S_kailasa spectro_kailasa(:)];
clear audio_kailasa spectro_kailasa;
end
clear a;
end
%%
% data = [fourier_blue fourier_eminem];
% data = [audio_blue audio_eminem];

```

```

data = [S_alizee(:,1:110) S_eminem(:,1:110) S_kailasa(:,1:110)];

% [m,n] = size(data);
% mn = mean(data,2);
% data = data - repmat(mn,1,n);
[u,s,v] = svd(data,'econ');

v_alizee = v(1:110,:);
v_eminem = v(111:220,:);
v_kailasa = v(221:end,:);
% v_test_blue = (inv(s)*u'*S_blue(:,91:end))';
% v_test_eminem = (inv(s)*u'*S_eminem(:,91:100))';
%%
% for l=1:100
%     m = 4; n = 39;
% for m=1:330
% for n=m:330
q1 = randperm(110); q2 = randperm(110); q3 = randperm(110);
train = [v_alizee(q1(1:100),m:n); v_eminem(q2(1:100),m:n);
v_kailasa(q3(1:100),m:n)];
% train = [v_alizee(:,m:n); v_eminem(:,m:n)];
test = [v_alizee(q1(101:end),m:n); v_eminem(q2(101:end),m:n);
v_kailasa(q3(101:end),m:n)];
% test = [v_test_alizee(:,m:n); v_test_eminem(:,m:n)];
label = [ones(100,1); 2*ones(100,1); 3*ones(100,1)];
% label = [ones(90,1); 2*ones(90,1)];

% gm = fitgmdist(train,2,'CovarianceType','diagonal');
% pre = cluster(gm,test);
% figure; bar(pre)

nb = fitcnb(train,label);
pre = nb.predict(test);
bar(pre)

% pre = classify(test,train,label,'diagquadratic');
% bar(pre)

% svm = svmtrain(train,label);
% pre = svmclassify(svm,test);
% figure; bar(pre);

actual = [ones(10,1); 2*ones(10,1); 3*ones(10,1)];
test_accuracy = (sum((actual-pre)==0)/30)*100;

% actual = [ones(10,1); 2*ones(10,1); 3*ones(10,1)];
% train_accuracy(1,1) = (sum((actual-pre)==0)/30)*100;

% end
% end
% end
% mean(train_accuracy)
% max(test_accuracy)
% 7-16, 4-21, 4-39 83.3333%

```