Homework 3: Music Genre Identification
AMATH 482: Computational Methods for Data Analysis, Winter 2017

Sean Lam
seanlam@uw.edu

February 20, 2017

Abstract

This report investigates the application of the Linear Discriminant Analysis classification algorithm to differentiate between music genres. This algorithm uses a probabilistic approach to a sample training data set to classify new cases/data. The results show the simplicity and robustness of the LDA algorithm.

I. Introduction and Overview

Machine learning is a relatively new field which has grown to become very popular and powerful because of its ability to perform a variety of tasks that could previously only be done by humans. The field focuses on the development of computer programs that can adapt to new data, and learn to make decisions as accurately as possible without being explicitly programmed. The power of machine learning comes from the ability to find crucial features and patterns in data sets without making any hypotheses. Machine learning has made a significant impact in many fields, such as finance, search engines, speech recognition, computer vision, optimization, and medicine.

There are two main types of machine learning algorithms: unsupervised and supervised algorithms. Unsupervised algorithms draw inferences from datasets without any prior labeling of input data. An example of this is clustering, where the algorithm finds patterns and trends in data and separates them into different categories. Supervised algorithms return predictions based on labeled training sets. In this project, the supervised Linear Discriminant Analysis algorithm was used to differentiate between different music bands and genres. It was trained with a random sample of songs and then cross validated with a test set to verify the robustness of the algorithm.

II. Theoretical Background

The Fourier transform takes a function of time and transforms into the frequency domain, in other words, decomposing the function into the frequencies (sins and cosines) that make it up.
The Fourier transform over the entire line $x \in (-\infty, \infty)$ is defined as:

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

The drawback of the Fourier transform is that it eliminates all time-domain information since it integrates over all time. This means that it fails to capture the moment in time when various frequencies were actually exhibited. This is useless for music because music is made up of varying frequencies over time. The solution to this problem is to split the signal into small windows of time. For each time window, the Fourier transform is then applied to characterize all the signals present in that time frame.

To account for both the time and frequency content, the Fourier transform kernel is modified with the addition of a window that can localize both time and frequency. This window was introduced by Gábor Dénes:

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

This results in the Gábor transform, or the short-time Fourier transform (STFT).

$$G[f](t, \omega) = \tilde{f}_g(t, \omega) = \int_{-\infty}^{\infty} f(\tau) \bar{g}(t - \tau) \mathrm{e}^{-i\omega\tau} d\tau \tag{3}$$

The Gábor method sacrifices some measure of accuracy in both the time and frequency domain in order to give both time and frequency resolution simultaneously. Similar to the Heisenberg uncertainty principle, the smaller the time window, the less information there is about the frequency content, and vice versa; wider time windows retain more frequency content but loses accuracy on the time resolution on the signal.

The wavelet method is a way to improve the accuracy and resolution of the Gábor transform. The idea is to vary the scaling window a. The low-frequency (low time resolution) components are extracted with a broad scaling window, and the high-frequency (high time resolution) components are extracted with a small scaling window.

Wavelet analysis begins with the function known as the mother wavelet with width parameter a and translation parameter b:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{4}$$

The most basic Gábor window is a Gaussian time filter centered at time τ with width a with translation parameter b.

$$g(t) = e^{-a(t-b)^2} \tag{5}$$

The singular value decomposition (SVD) takes a matrix and decomposes it into three constitutive components. It takes the form:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\star \tag{6}$$

with the following three matrices:

$$\mathbf{U} \in \mathbb{C}^{m\times m} \text{ is unitary} \tag{7}$$

$$\mathbf{V} \in \mathbb{C}^{n\times n} \text{ is unitary} \tag{8}$$

$$\mathbf{\Sigma} \in \mathbb{R}^{m\times n} \text{ is diagonal} \tag{9}$$

The diagonal entries of Σ are nonnegative and are ordered from largest to smallest.

The $\mathbf{U}$ matrix contains all of the principal components (orthogonal bases) for the dataset. In this case, it is the principal characteristics of all of the songs. The $\mathbf{\Sigma}$ diagonal matrix shows how "important" each of the principal components are, which can be used to find the dominant modes that can represent the majority of the original dataset. The $\mathbf{V}$ matrix contains the features of all of the songs. It shows how much each song projects onto the principal components of $\mathbf{U}$.

The supervised classification algorithm used in this project was the linear discriminant analysis (LDA). The goal of the LDA is to find a suitable projection that maximizes the between the inter-class data while maximizing the intra-class data.

This report uses the three class LDA, but it follows the same concepts as the two class LDA.

Construct a projection $\mathbf{w}$ such that

$$\mathbf{w} = argmax_{\mathbf{w}} \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}} \tag{10}$$

where the scatter matrices for between-class $\mathbf{S}_B$ and within-class $\mathbf{S}_W$ data are given by

$$\mathbf{S}_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T \tag{11}$$

$$\mathbf{S}_W = \sum_{j=1}^{2}\sum_{x}(\mathbf{x} - \mu_j)(\mathbf{x} - \mu_j)^T \tag{12}$$

The solution can be found by the eigenvalue problem

$$\mathbf{S}_B\mathbf{w} = \lambda\mathbf{S}_W\mathbf{w} \tag{13}$$

The first step is to feed the algorithm with a training set of features, data that is already labeled, to construct the optimal projection basis which separates the data most effectively. The features are taken from the $\mathbf{V}$ matrix found earlier from the singular value decomposition. Then when a new set of data is inputted, the algorithm can classify it to a class based on the projection found earlier with the training set.

III. Algorithm Implementation and Development

For each test, I chose to use five songs for each of three artists. After collecting .mp3 files of the songs, I converted them to matrix form using the audioread command. The resulting matrix is two dimensional, N x 2 (N being number of frames). The two columns are the two data channels (left and right speakers), so I added both of them together to get a column vector. I had considered just eliminating one of the channels, but I got less accuracy that way. Next, the beginning and end of each song had some silence (amplitude of less than 0.005), so I went and truncated those out.

Next, the algorithm determined what the maximum number of five second clips I could sample from the shortest song, because I wanted to sample an equal number of clips per song. The algorithm then split the song into five second clips.

The next step was to perform a Gábor transform on each clip in order to make a spectrogram of each clip. First, I downsampled each song so I would take every fifth frame, which sped up computational time but still retained most of the data integrity. I decided to use a Gaussian filter because it was the simplest. Different parameters for the window were experimented, but a value of 500 was chosen based on how the spectrogram looked (approximately equal amount of frequency and time resolution).

The algorithm to compute the spectrogram was for every clip:

1. Slide window t_{slide} seconds
2. Generate Gábor filter
3. Perform element-wise multiplication on the filter and the data
4. Perform fast Fourier transform on this product
5. Place the zero-frequency centered (fftshift) vector to next row of spectrogram

All of the spectrograms were reshaped into column vectors, and concatenated into a big matrix containing all the spectrograms in the data set.
The matrix X_{spec} is size N $\times$ M

$$X_{spec} = \begin{bmatrix} spec_1 & spec_2 & spec_3 & ... & spec_m \\ ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... \end{bmatrix} \tag{14}$$

The economy singular value decomposition of the spectrogram matrix taken. The first $\frac{1}{3}$ of the rows of $\mathbf{V}$ contained the features of the first artist. The second third contained the rows of the second artist, the the last third contained the rows of the third artist. The columns of the $\mathbf{V}$ matrix were the features, sorted from most important to least important (highest to lowest σ values). I used the first 40 features of $\mathbf{V}$.

The last step was to set up the LDA classification algorithm. We have N songs, so each artist has $\frac{N}{3}$ features. I then generated three vectors of random numbers from 1 to $\frac{N}{3}$, so I could use it to pick out random samples of each artist. After extracting random rows from the $\mathbf{V}$ matrix, I created a training data set using the first 60% of the random features. The test set would then be the remaining 40% of the features. Then I ran the LDA algorithm, and plotted the results on a bar graph.

IV. Computational Results

The first test was to have the algorithm differentiate between artists of different genres. I chose to do N Sync (Boy Band), Yo-Yo Ma (Cello), and Ice Cube (Rap). 26 samples were chosen for each song (five total songs per artist). Figure 1 shows the spectrograms of one of each artist. It can be seen clearly how different the songs are. The Yo-Yo Ma sample (Figure 1a) has the smallest frequency range, which is expected because this sample only contains cello, without other background music. The N Sync sample (Figure 1b) has a medium frequency range because it has a relatively medium beat and frequency. The Ice Cube sample (Figure 1c) has a large frequency range because it has a low frequency beat, with medium frequency vocals and higher frequency sounds thrown in the mix.

The first three modes of Test 1 are plotted on Figure 2a. We can compare the the analysis from the spectrograms to the modes by seeing the the Yo-Yo Ma modes are more compacted into one small place, the N Sync modes are a little more spread out, and the Ice Cube modes cover a wide range. A total of forty modes were used for the classification algorithm. Figure 2b shows the results of the LDA classification. This run had an accuracy of over 98%, but an average taken over 100 runs was around 96%.

The second test was to have the algorithm differentiate between artists of the same genre. I chose to do three boy bands: N Sync, Backstreet Boys, and One Direction. 34 samples were chosen for each song. Figure 3 shows the

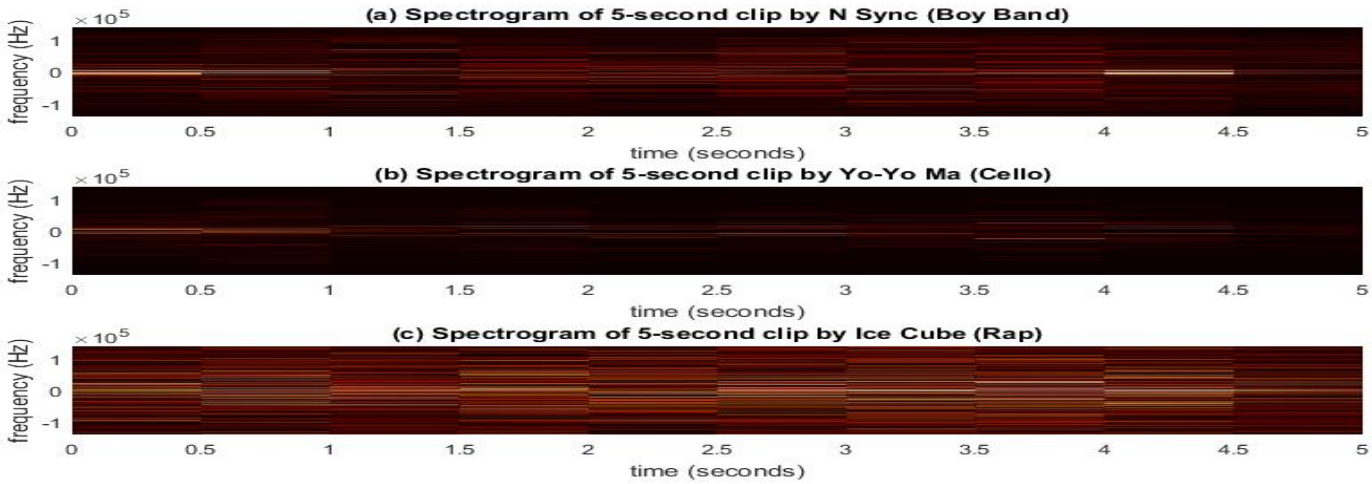

Figure 1: These are spectrograms of 5-second samples of each artist in Test 1. (a) shows the spectrogram of a N Sync sample, (b) shows the spectrogram of a Yo-Yo Ma sample, (c) shows the spectrogram of an Ice Cube sample.

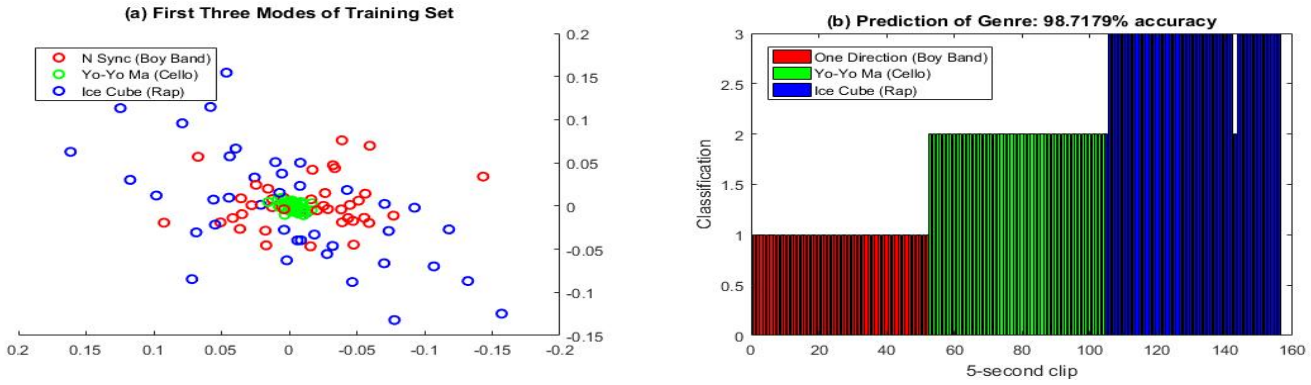

Figure 2: These are the results for Test 1. (a) shows the first three SVD modes of the training set, and (b) shows the classification result with 98.7179% accuracy. (Classifiers: Red = 1, Green = 2, Blue = 3)

spectrograms of Test 2. They look relatively similar, which is expected, because they are of the same genre (similar tempo, frequencies, range, etc).

The first three modes of Test 2 are plotted on Figure 4a. As we expected from the spectrogram, the modes are not as neatly separated as they were in Test 1. The N Sync and Backstreet Boys modes are spread out pretty evenly, while the One Direction modes are more compact. Figure 4b shows the results of the LDA classification. The classifier classifies N Sync as the Backstreet Boys sometimes and vice versa, while the One Direction predictions were mostly spot on. This can be explained by the similar spread of modes for N Sync and Backstreet Boys as shown in Figure 4a. This particular run had an accuracy of 90.1961%, but an average taken over 100 runs was around 88%. This is about 8% lower than the accuracy in Test 1, which is expected, because artists of the same genre sound very similar to each other.

The third test was to have the algorithm differentiate between different genres of music. I used five different artists for each genre to generalize it as much as possible. The genres I chose were boy band, classical, and rap. 39 samples were chosen for each song. Figure 5 shows the spectrograms of Test 3. We can see from the spectrograms that classical music has a smaller frequency range compared to boy bands and rap.

The first three modes of Test 3 are plotted on Figure 6a. As seen from the spectrograms, classical music covers a really small frequency range in comparison to boy band and rap. The boy band and rap modes both stretch out in a rectangular shape but at different angles. Figure 6b shows the results of the LDA classification. The classifier had no trouble classifying classical music, but sometimes was wrong for boy band and rap, which is expected because they both have a wide range of frequencies. This particular run had an accuracy of 95.7265%, but an average taken over 100 runs was around 93%.

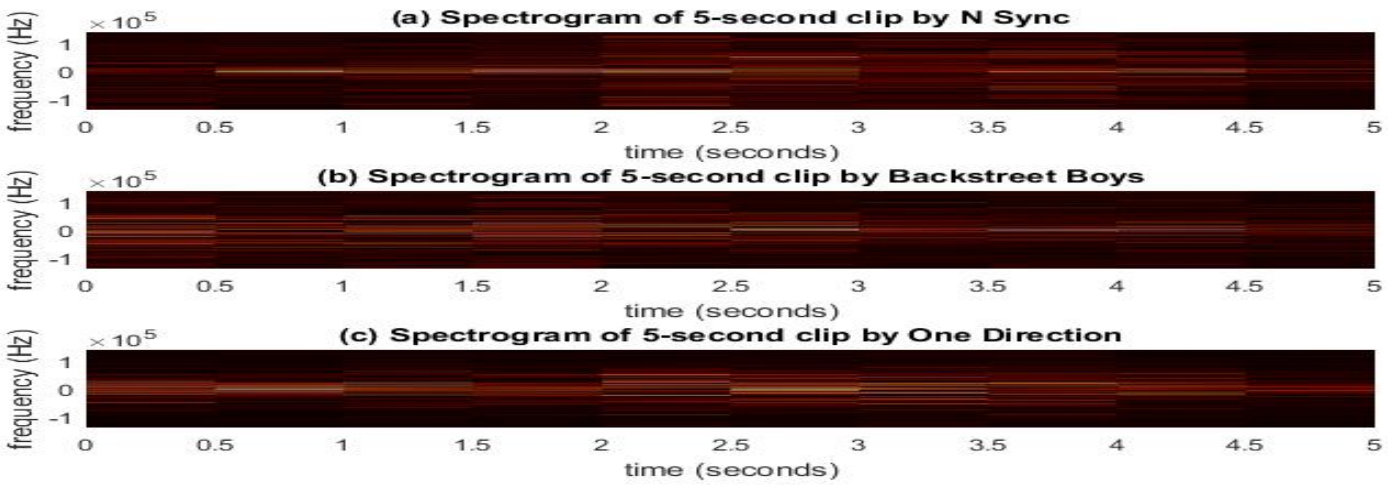

Figure 3: These are spectrograms of 5-second samples of each artist in Test 2. (a) shows the spectrogram of a N Sync sample, (b) shows the spectrogram of a Backstreet Boys sample, (c) shows the spectrogram of a One Direction sample.

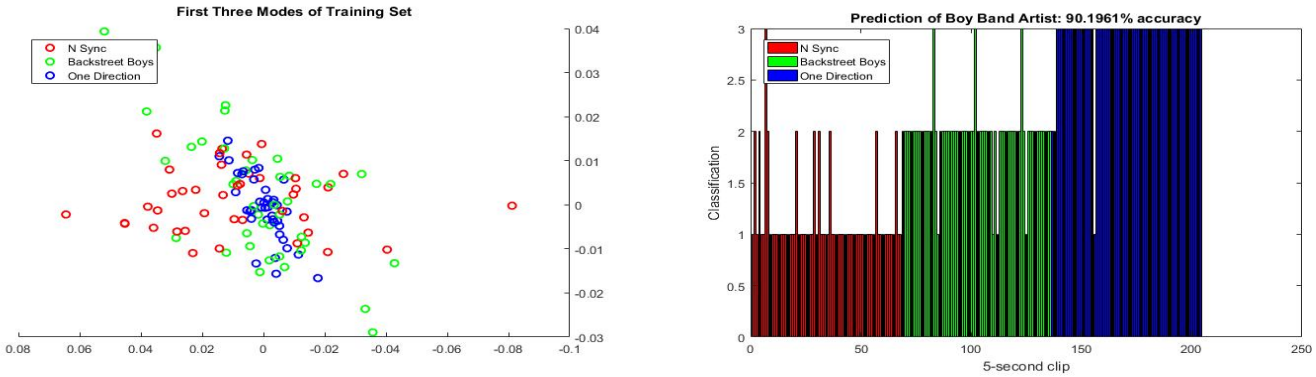

Figure 4: These are the results for Test 2. (a) shows the first three SVD modes of the training set, and (b) shows the classification results with 90.1961% accuracy. (Classifiers: Red = 1, Green = 2, Blue = 3)

V. Summary and Conclusions

In this report we investigated the effectiveness of the linear discriminant analysis algorithm for classification. In Test 1, LDA was able to differentiate between artists of different genres with over 96% accuracy. In Test 2, LDA was able to differentiate between artists of the same genres with over 88% accuracy. In Test 3, LDA was able to differentiate between genres of music with over 93% accuracy. The trend of these results were expected, because Test 1 was differentiating artists of different genres, which should be the easiest to do, and Test 3 was differentiating different genres, which should be the next easiest, and Test 2 should be the hardest, because it was differentiating between artists of the same genre, which is pretty hard even to do by ear. It was really surprising how accurate test 2 was, considering how similar music of the same genre sounds like.

Classification is one of the simplest concepts in machine learning, and is necessary in every field of data science. This report shows the robustness of the LDA classification algorithm and how easy it is to implement it into any project. The difficult part is scrubbing and organizing raw data into the right form so that the algorithm can produce significant results. Cross validation is also an important concept; it is necessary that the training set is not too big so that the data is overfitted.

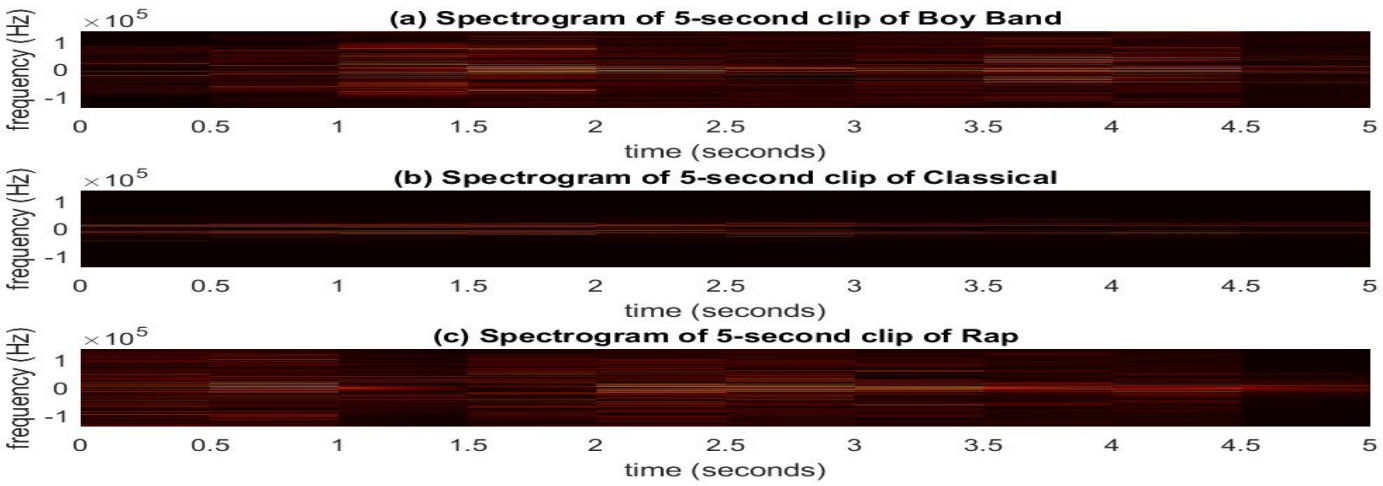

Figure 5: These are spectrograms of 5-second samples of each artist in Test 3. (a) shows the spectrogram of a Boy Band sample, (b) shows the spectrogram of a Classical sample, (c) shows the spectrogram of a Rap sample.

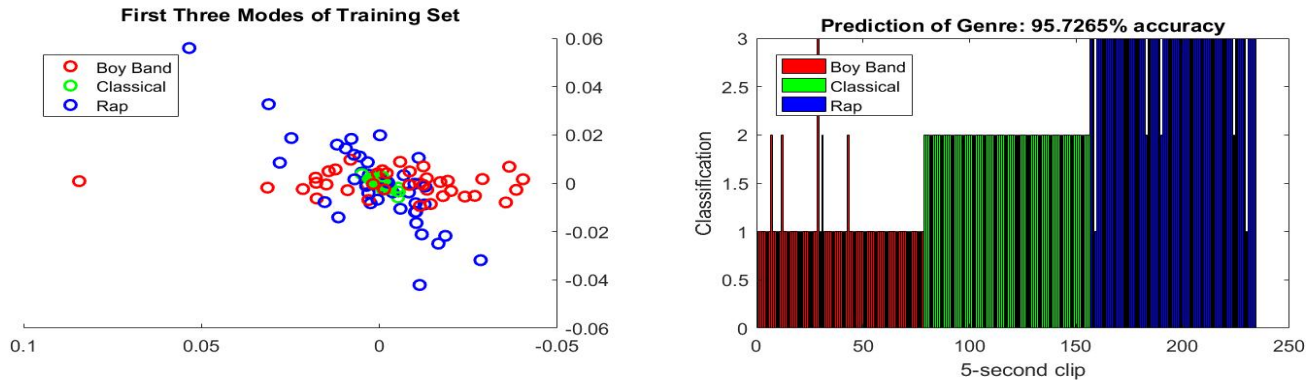

Figure 6: These are the results for Test 3. (a) shows the first three SVD modes of the training set, and (b) shows the classification results with 95.7265% accuracy. (Classifiers: Red = 1, Green = 2, Blue = 3)

Appendix A: MATLAB functions used and brief implementation explanation

[y, FS] = audioread(filename) : Reads an audio file, returns the sampled data y and sample rate FS (in Hz)
y = downsample(y, n) : Reads every nth point of the vector y, and returns the new vector
y = fftshift(x) : takes in an input x and performs the fast Fourier transform, and shifts it so its zero frequency component is in the center of the spectrum
y = fft(x) : computes the discrete Fourier transform of x and returns it as y
[U, S, V] = svd(A) : performs singular value decomposition on matrix A and outputs $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$
r = randperm(n) : Creates a vector of length n with random numbers from 1-n, never repeating numbers
predict = classify(test, train, classifier) : performs linear discriminant analysis on a test set with the given training set and classifier, and outputs the predictions
song = Truncate(RawSong) : takes in an audioread mp3 file, eliminates frames with values near zero (for truncating beginning and end silence)
[spec, ks, tslide] = makeSpectrogram(song) : inputs a sample of music and returns the Gábor transform spectrogram, gaussian window, and time slider
avgaccuracy = AverageResult(modes, numOfEach, V, cycles) : runs classification algorithm (cycles) number of times, with (modes) number of features of (V), (numOfEach) samples of each case, outputs the average accuracy of classification algorithm

Appendix B: MATLAB codes

Test1.m

```
[Raw1,FS] = audioread('NSYNC_ByeByeBye.mp3');
[Raw2,~] = audioread('NSYNC_ItsGonnaBeMe.mp3');
[Raw3,~] = audioread('NSYNC_IWantYouBack.mp3');
[Raw4,~] = audioread('NSYNC_Pop.mp3');
[Raw5,~] = audioread('NSYNC_TearingUpMyHeart.mp3');
[Raw6,~] = audioread('YoYoMa_BachCelloSuite1Menuet.mp3');
[Raw7,~] = audioread('YoYoMa_BachCelloSuite1Prelude.mp3');
[Raw8,~] = audioread('YoYoMa_BachCelloSuite2Gigue.mp3');
[Raw9,~] = audioread('YoYoMa_BachCelloSuite2Menuet.mp3');
[Raw10,~] = audioread('YoYoMa_BachCelloSuite2Prelude.mp3');
[Raw11,~] = audioread('IceCube_Friday.mp3');
[Raw12,~] = audioread('IceCube_CheckYoSelf.mp3');
[Raw13,~] = audioread('IceCube_IRepThatWest.mp3');
[Raw14,~] = audioread('IceCube_ItWasAGoodDay.mp3');
[Raw15,~] = audioread('IceCube_WhyWeThugs.mp3');

%% eliminate beginning zeros in each video
% add left and right channels together
song1 = Truncate(Raw1);
song2 = Truncate(Raw2);
song3 = Truncate(Raw3);
song4 = Truncate(Raw4);
song5 = Truncate(Raw5);
song6 = Truncate(Raw6);
song7 = Truncate(Raw7);
song8 = Truncate(Raw8);
song9 = Truncate(Raw9);
song10 = Truncate(Raw10);
song11 = Truncate(Raw11);
song12 = Truncate(Raw12);
song13 = Truncate(Raw13);
song14 = Truncate(Raw14);
song15 = Truncate(Raw15);
songs = {song1, song2, song3, song4, song5, song6, song7, song8, song9,
song10, song11, song12, song13, song14, song15};

%% find number of 5 second clips to collect
shortestTime = 10^20;
for j = 1:length(songs)
    current = cell2mat(songs(j));
    shortestTime = min(shortestTime, length(current));
end
clipsize = 5 * FS;
numClips = floor(shortestTime / clipsize);

%% split into clips
clipsize = 5 * FS;
X = [];
for j = 1:length(songs)
   currentSong = cell2mat(songs(j)); %get song
   for k = 1:numClips
```

```
       currentClip = currentSong(1+clipsize*(k-1):clipsize*k); %take clip
       X = [X currentClip]; %add it to main matrix
   end
end

%% Gabor transform / spectrogram
Xspec = [];
for j = 1:(numClips*length(songs))
   current = X(:,j);
   [currentspec] = makeSpectrogram(current);
   [a, b] = size(currentspec);
   currentspec = reshape(currentspec, a*b, 1);
   Xspec = [Xspec currentspec];
end


%% SVD
[U,S,V] = svd(Xspec, 'econ');
percentS = diag(S) / sum(diag(S)) * 100;


%% machine learning
numOfEach = floor(length(S)/3);
rand1 = randperm(numOfEach);
rand2 = randperm(numOfEach);
rand3 = randperm(numOfEach);
modes = 40;
x1 = V(1:numOfEach, 1:modes);
x2 = V(numOfEach+1:numOfEach*2, 1:modes);
x3 = V(numOfEach*2+1:numOfEach*3, 1:modes);
numToTrain = floor(numOfEach*.6);
xtrain = [x1(rand1(1:numToTrain),:) ;
    x2(rand2(1:numToTrain),:) ;
    x3(rand3(1:numToTrain),:) ];
figure(1)
subplot(1,2,1)
plot3(x1(2,:),x1(3,:),x1(4,:),'ro','Linewidth',[1.5])
hold on
plot3(x2(2,:),x2(3,:),x2(4,:),'go','Linewidth',[1.5])
hold on
plot3(x3(2,:),x3(3,:),x3(4,:),'bo','Linewidth',[1.5])
title('(a) First Three Modes of Training Set')
legend('N Sync (Boy Band)', 'Yo-Yo Ma (Cello)', 'Ice Cube (Rap)', 'Location', 'Northwest')
xtest = [x1(rand1(numToTrain+1:end),:) ;
    x2(rand2(numToTrain+1:end),:) ;
    x3(rand3(numToTrain+1:end),:) ];
c1 = [ones(numToTrain,1);2*ones(numToTrain,1); 3*ones(numToTrain,1)];
pre = classify(xtest, xtrain, c1);
numToTest = numOfEach - numToTrain;
counter = 0;
for j = 1:numToTest
    if (pre(j) == 1)
       counter = counter + 1;
    end
end
for j = numToTest+1:numToTest*2
   if (pre(j) == 2)
```

```
       counter = counter + 1;
   end
end
for j = numToTest*2+1:numToTest*3
    if (pre(j) == 3)
        counter = counter + 1;
    end
end
accuracy = counter / numToTest/3 * 100;
subplot(1,2,2)
bar(1:numToTest,pre(1:numToTest),'r')
hold on
bar(numToTest+1:numToTest*2,pre(numToTest+1:numToTest*2),'g')
hold on
bar(numToTest*2+1:numToTest*3,pre(numToTest*2+1:numToTest*3),'b')
xlabel('5-second clip')
ylabel('Classification')
title('(b) Prediction of Genre: ' + string(accuracy) + '% accuracy')
legend('One Direction (Boy Band)', 'Yo-Yo Ma (Cello)','Ice Cube (Rap)','Location','Northwest')

%% cross validate
modes = 40;
cycles = 100;
accuracy = AverageResult(modes, numOfEach, V, cycles)
```

Test2.m

```
[Raw1,FS] = audioread('NSYNC_ByeByeBye.mp3');
[Raw2,~] = audioread('NSYNC_ItsGonnaBeMe.mp3');
[Raw3,~] = audioread('NSYNC_IWantYouBack.mp3');
[Raw4,~] = audioread('NSYNC_Pop.mp3');
[Raw5,~] = audioread('NSYNC_TearingUpMyHeart.mp3');
[Raw6,~] = audioread('BackstreetBoys_AsLongAsYouLoveMe.mp3');
[Raw7,~] = audioread('BackstreetBoys_DontWannaLoseYouNow.mp3');
[Raw8,~] = audioread('BackstreetBoys_IWantItThatWay.mp3');
[Raw9,~] = audioread('BackstreetBoys_ShapeOfMyHeart.mp3');
[Raw10,~] = audioread('BackstreetBoys_ShowMeTheMeaningOfBeingLonely.mp3');
[Raw11,~] = audioread('OneDirection_LittleThings.mp3');
[Raw12,~] = audioread('OneDirection_LiveWhileWereYoung.mp3');
[Raw13,~] = audioread('OneDirection_NightChanges.mp3');
[Raw14,~] = audioread('OneDirection_StoryofMyLife.mp3');
[Raw15,~] = audioread('OneDirection_YouAndI.mp3');

%% eliminate beginning and end zeros in each video
% add left and right channels together
song1 = Truncate(Raw1);
song2 = Truncate(Raw2);
song3 = Truncate(Raw3);
song4 = Truncate(Raw4);
song5 = Truncate(Raw5);
song6 = Truncate(Raw6);
song7 = Truncate(Raw7);
song8 = Truncate(Raw8);
song9 = Truncate(Raw9);
song10 = Truncate(Raw10);
song11 = Truncate(Raw11);
song12 = Truncate(Raw12);
song13 = Truncate(Raw13);
```

```
song14 = Truncate(Raw14);
song15 = Truncate(Raw15);

songs = {song1, song2, song3, song4, song5, song6, song7, song8, song9,
song10, song11, song12, song13, song14, song15};


%% find number of 5 second clips to collect
shortestTime = 10^20;
for j = 1:length(songs)
    current = cell2mat(songs(j));
    shortestTime = min(shortestTime, length(current));
end
clipsize = 5 * FS;
numClips = floor(shortestTime / clipsize);

%% split into clips
clipsize = 5 * FS;
X = [];
for j = 1:length(songs)
   currentSong = cell2mat(songs(j)); %get song
   for k = 1:numClips
       currentClip = currentSong(1+clipsize*(k-1):clipsize*k); %take clip
       %currentClip = downsample(currentClip,2); %take every other point
       X = [X currentClip]; %add it to main matrix
   end
end

%% Gabor transform / spectrogram
Xspec = [];
for j = 1:(numClips*length(songs))
   current = X(:,j);
   [currentspec] = makeSpectrogram(current);
   [a, b] = size(currentspec);
   currentspec = reshape(currentspec, a*b, 1);
   Xspec = [Xspec currentspec];
   j
end

%% SVD
[U,S,V] = svd(Xspec, 'econ');
percentS = diag(S) / sum(diag(S)) * 100;

%% machine learning
numOfEach = floor(length(S) / 3);
rand1 = randperm(numOfEach);
rand2 = randperm(numOfEach);
rand3 = randperm(numOfEach);
modes = 40;
x1 = V(1:numOfEach, 1:modes);
x2 = V(numOfEach+1:numOfEach*2, 1:modes);
x3 = V(numOfEach*2+1:numOfEach*3, 1:modes);
numToTrain = floor(numOfEach*.6);
x1train = x1(rand1(1:numToTrain),:);
x2train = x2(rand2(1:numToTrain),:);
x3train = x3(rand3(1:numToTrain),:);
xtrain = [ x1train;
```

```
    x2train ;
    x3train ];
figure(1)
subplot(1,2,1)
plot3(x1(1,:),x1(2,:),x1(3,:),'ro','Linewidth',[1.5])
hold on
plot3(x2(1,:),x2(2,:),x2(3,:),'go','Linewidth',[1.5])
hold on
plot3(x3(1,:),x3(2,:),x3(3,:),'bo','Linewidth',[1.5])
title('First Three Modes of Training Set')
legend('N Sync', 'Backstreet Boys', 'One Direction', 'Location', 'Northwest')
xtest = [x1(rand1(numToTrain+1:end),:) ;
    x2(rand2(numToTrain+1:end),:) ;
    x3(rand3(numToTrain+1:end),:) ];
c1 = [ones(numToTrain,1);2*ones(numToTrain,1); 3*ones(numToTrain,1)];
%nb = fitNaiveBayes(xtrain, c1);
%predict = nb.predict(xtest);
predict = classify(xtest, xtrain, c1);
numToTest = numOfEach - numToTrain;
counter = 0;
for j = 1:numToTest
    if (predict(j) == 1)
       counter = counter + 1;
    end
end
for j = numToTest+1:numToTest*2
   if (predict(j) == 2)
       counter = counter + 1;
   end
end
for j = numToTest*2+1:numToTest*3
    if (predict(j) == 3)
        counter = counter + 1;
    end
end
accuracy = counter / numToTest/3 * 100;
subplot(1,2,2)
bar(1:numToTest,predict(1:numToTest),'r')
hold on
bar(numToTest+1:numToTest*2,predict(numToTest+1:numToTest*2),'g')
hold on
bar(numToTest*2+1:numToTest*3,predict(numToTest*2+1:numToTest*3),'b')
xlabel('5-second clip')
ylabel('Classification')
title('Prediction of Boy Band Artist: ' + string(accuracy) + '% accuracy')
legend('N Sync', 'Backstreet Boys','One Direction','Location','Northwest')

%% cross validate
modes = 40;
cycles = 100;
accuracy = AverageResult(modes, numOfEach, V, cycles)
```

Test3.m

```
[Raw1,FS] = audioread('NSYNC_ByeByeBye.mp3');
[Raw2,~] = audioread('BackstreetBoys_IWantItThatWay.mp3');
[Raw3,~] = audioread('BigTimeRush_TillIForgetAboutYou.mp3');
[Raw4,~] = audioread('JonasBrothers_WhenYouLookMeInTheEyes.mp3');
```

```
[Raw5,~] = audioread('OneDirection_StoryofMyLife.mp3');
[Raw6,~] = audioread('ClaudeDebussy_ClairDeLune.mp3');
[Raw7,~] = audioread('Tchaikovsky_WaltzoftheFlowers.mp3');
[Raw8,~] = audioread('Albinoni_AdagioInG Minor.mp3');
[Raw9,~] = audioread('Bach_Air.mp3');
[Raw10,~] = audioread('Bach_OrchestralSuiteNo3DdurBWV1068.mp3');
[Raw11,~] = audioread('2pac_Changes.mp3');
[Raw12,~] = audioread('EazyE-StillANgga.mp3');
[Raw13,~] = audioread('IceCube_CheckYoSelf.mp3');
[Raw14,~] = audioread('KendrickLamar_MoneyTrees.mp3');
[Raw15,~] = audioread('NWA_StraightOuttaCompton.mp3');

%% eliminate beginning and end zeros in each video
% add left and right channels together
song1 = Truncate(Raw1);
song2 = Truncate(Raw2);
song3 = Truncate(Raw3);
song4 = Truncate(Raw4);
song5 = Truncate(Raw5);
song6 = Truncate(Raw6);
song7 = Truncate(Raw7);
song8 = Truncate(Raw8);
song9 = Truncate(Raw9);
song10 = Truncate(Raw10);
song11 = Truncate(Raw11);
song12 = Truncate(Raw12);
song13 = Truncate(Raw13);
song14 = Truncate(Raw14);
song15 = Truncate(Raw15);

songs = {song1, song2, song3, song4, song5, song6, song7, song8, song9,
song10, song11, song12, song13, song14, song15};


%% find number of 5 second clips to collect
shortestTime = 10^20;
for j = 1:length(songs)
    current = cell2mat(songs(j));
    shortestTime = min(shortestTime, length(current));
end
clipsize = 5 * FS;
numClips = floor(shortestTime / clipsize);

%% split into clips
clipsize = 5 * FS;
X = [];
for j = 1:length(songs)
   currentSong = cell2mat(songs(j)); %get song
   for k = 1:numClips
       currentClip = currentSong(1+clipsize*(k-1):clipsize*k); %take clip
       %currentClip = downsample(currentClip,2); %take every other point
       X = [X currentClip]; %add it to main matrix
   end
end

%% Gabor transform / spectrogram
Xspec = [];
```

```
for j = 1:(numClips*length(songs))
   current = X(:,j);
   [currentspec] = makeSpectrogram(current);
   [a, b] = size(currentspec);
   currentspec = reshape(currentspec, a*b, 1);
   Xspec = [Xspec currentspec];
   j
end

%% SVD
[U,S,V] = svd(Xspec, 'econ');
percentS = diag(S) / sum(diag(S)) * 100;

%% machine learning
numOfEach = floor(length(S) / 3);
rand1 = randperm(numOfEach);
rand2 = randperm(numOfEach);
rand3 = randperm(numOfEach);
modes = 40;
x1 = V(1:numOfEach, 1:modes);
x2 = V(numOfEach+1:numOfEach*2, 1:modes);
x3 = V(numOfEach*2+1:numOfEach*3, 1:modes);
numToTrain = floor(numOfEach*.6);
x1train = x1(rand1(1:numToTrain),:);
x2train = x2(rand2(1:numToTrain),:);
x3train = x3(rand3(1:numToTrain),:);
xtrain = [ x1train;
    x2train ;
    x3train ];
figure(1)
subplot(1,2,1)
plot3(x1(1,:),x1(2,:),x1(3,:),'ro','Linewidth',[1.5])
hold on
plot3(x2(1,:),x2(2,:),x2(3,:),'go','Linewidth',[1.5])
hold on
plot3(x3(1,:),x3(2,:),x3(3,:),'bo','Linewidth',[1.5])
title('First Three Modes of Training Set')
legend('Boy Band', 'Classical', 'Rap', 'Location', 'Northwest')
xtest = [x1(rand1(numToTrain+1:end),:) ;
    x2(rand2(numToTrain+1:end),:) ;
    x3(rand3(numToTrain+1:end),:) ];
c1 = [ones(numToTrain,1);2*ones(numToTrain,1); 3*ones(numToTrain,1)];
%nb = fitNaiveBayes(xtrain, c1);
%predict = nb.predict(xtest);
predict = classify(xtest, xtrain, c1);
numToTest = numOfEach - numToTrain;
counter = 0;
for j = 1:numToTest
    if (predict(j) == 1)
       counter = counter + 1;
    end
end
for j = numToTest+1:numToTest*2
   if (predict(j) == 2)
       counter = counter + 1;
   end
end
```

```
for j = numToTest*2+1:numToTest*3
    if (predict(j) == 3)
        counter = counter + 1;
    end
end
accuracy = counter / numToTest/3 * 100;
subplot(1,2,2)
bar(1:numToTest,predict(1:numToTest),'r')
hold on
bar(numToTest+1:numToTest*2,predict(numToTest+1:numToTest*2),'g')
hold on
bar(numToTest*2+1:numToTest*3,predict(numToTest*2+1:numToTest*3),'b')
xlabel('5-second clip')
ylabel('Classification')
title('Prediction of Genre: ' + string(accuracy) + '% accuracy')
legend('Boy Band', 'Classical','Rap','Location','Northwest')

%% cross validate
modes = 40;
cycles = 100;
accuracy = AverageResult(modes, numOfEach, V, cycles)
```

Truncate.m

```
%eliminates all frames with value near zero (this truncates beginning and
%end of song that is mostly quiet)
%also adds left and right channels together
function truncateSong = Truncate(rawSong)
    indices = find(abs(rawSong(:,1) > 0.005));
    truncateSongL = rawSong(indices(1):indices(end),1);
    truncateSongR = rawSong(indices(1):indices(end),2);
    truncateSong = truncateSongL + truncateSongR;
end
```

makeSpectrogram.m

```
function [spec, ks, tslide] = makeSpectrogram(song)
    song = song';
    song = downsample(song,5);
    FS = 44100;
    L = length(song) / FS;
    n = length(song);
    t2 = linspace(0,L,n+1);
    t = t2(1:n);
    k = (2*pi/L)*[0:n/2 -n/2:-1];
    ks = fftshift(k);
    tslide = 0:0.1:L;

    spec=[];
    a = 500;
    for j=1:length(tslide)
        g=exp(-a*(t-tslide(j)).^2); % Gabor
        Sg=g.*song; Sgt=fft(Sg);
        spec=[spec; abs(fftshift(Sgt))];
    end
    %plot spectrogram
%     figure(1)
%     pcolor(tslide * 5, ks(1:end-1), spec.'), shading interp
%     colormap(hot)
```

```
%     title('Spectrogram of 5-second clip by Ice Cube')
%     xlabel('time (seconds)')
%     ylabel('frequency (Hz)')
end
```

AverageResult.m

```
function avgaccuracy = AverageResult(modes, numOfEach, V, cycles)
    x1 = V(1:numOfEach, 1:modes);
    x2 = V(numOfEach+1:numOfEach*2, 1:modes);
    x3 = V(numOfEach*2+1:numOfEach*3, 1:modes);
    numToTrain = floor(numOfEach*.6);
    numToTest = numOfEach - numToTrain;
    for k = 1:cycles
        rand1 = randperm(numOfEach);
        rand2 = randperm(numOfEach);
        rand3 = randperm(numOfEach);
        x1train = x1(rand1(1:numToTrain),:);
        x2train = x2(rand2(1:numToTrain),:);
        x3train = x3(rand3(1:numToTrain),:);
        xtrain = [ x1train;
            x2train ;
            x3train ];
        xtest = [x1(rand1(numToTrain+1:end),:) ;
            x2(rand2(numToTrain+1:end),:) ;
            x3(rand3(numToTrain+1:end),:) ];
        c1 = [ones(numToTrain,1);2*ones(numToTrain,1); 3*ones(numToTrain,1)];
        predict = classify(xtest, xtrain, c1);
        counter = 0;
        for j = 1:numToTest
            if (predict(j) == 1)
               counter = counter + 1;
            end
        end
        for j = numToTest+1:numToTest*2
           if (predict(j) == 2)
               counter = counter + 1;
           end
        end
        for j = numToTest*2+1:numToTest*3
            if (predict(j) == 3)
                counter = counter + 1;
            end
        end
        accuracy(k) = counter / numToTest /3 * 100;
    end
    avgaccuracy = sum(accuracy) / 100;
end
```

plotSpecs1.m

```
%plots spectrograms
figure(1)
subplot(3,1,1)
clipsize = 44100 * 5 - 1;
start = 1000000;
[spec, ks, tslide] = makeSpectrogram(song4(start:start + clipsize));
pcolor(tslide * 5, ks(1:end-1), spec.'), shading interp
colormap(hot)
```

```
title('(a) Spectrogram of 5-second clip by N Sync (Boy Band)')
xlabel('time (seconds)')
ylabel('frequency (Hz)')

subplot(3,1,2)
start = 500000;
[spec, ks, tslide] = makeSpectrogram(song7(start:start + clipsize));
pcolor(tslide * 5, ks(1:end-1), spec.'), shading interp
colormap(hot)
title('(b) Spectrogram of 5-second clip by Yo-Yo Ma (Cello) ')
xlabel('time (seconds)')
ylabel('frequency (Hz)')

subplot(3,1,3)
start = 2200000;
[spec, ks, tslide] = makeSpectrogram(song12(start:start + clipsize));
pcolor(tslide * 5, ks(1:end-1), spec.'), shading interp
colormap(hot)
title('(c) Spectrogram of 5-second clip by Ice Cube (Rap)')
xlabel('time (seconds)')
ylabel('frequency (Hz)')
```

plotSpecs2.m

```
%plots spectrograms
figure(1)
subplot(3,1,1)
clipsize = 44100 * 5 - 1;
start = 1000000;
[spec, ks, tslide] = makeSpectrogram(song3(start:start + clipsize));
pcolor(tslide * 5, ks(1:end-1), spec.'), shading interp
colormap(hot)
title('(a) Spectrogram of 5-second clip by N Sync')
xlabel('time (seconds)')
ylabel('frequency (Hz)')

subplot(3,1,2)
start = 5000000;
[spec, ks, tslide] = makeSpectrogram(song7(start:start + clipsize));
pcolor(tslide * 5, ks(1:end-1), spec.'), shading interp
colormap(hot)
title('(b) Spectrogram of 5-second clip by Backstreet Boys')
xlabel('time (seconds)')
ylabel('frequency (Hz)')

subplot(3,1,3)
start = 2200000;
[spec, ks, tslide] = makeSpectrogram(song13(start:start + clipsize));
pcolor(tslide * 5, ks(1:end-1), spec.'), shading interp
colormap(hot)
title('(c) Spectrogram of 5-second clip by One Direction')
xlabel('time (seconds)')
ylabel('frequency (Hz)')
```

plotSpecs3.m

```
%plots spectrograms
figure(1)
subplot(3,1,1)
```

```
clipsize = 44100 * 5 - 1;
start = 1200000;
[spec, ks, tslide] = makeSpectrogram(song2(start:start + clipsize));
pcolor(tslide * 5, ks(1:end-1), spec.'), shading interp
colormap(hot)
title('(a) Spectrogram of 5-second clip of Boy Band')
xlabel('time (seconds)')
ylabel('frequency (Hz)')

subplot(3,1,2)
start = 5000000;
[spec, ks, tslide] = makeSpectrogram(song6(start:start + clipsize));
pcolor(tslide * 5, ks(1:end-1), spec.'), shading interp
colormap(hot)
title('(b) Spectrogram of 5-second clip of Classical')
xlabel('time (seconds)')
ylabel('frequency (Hz)')

subplot(3,1,3)
start = 2200000;
[spec, ks, tslide] = makeSpectrogram(song13(start:start + clipsize));
pcolor(tslide * 5, ks(1:end-1), spec.'), shading interp
colormap(hot)
title('(c) Spectrogram of 5-second clip of Rap')
xlabel('time (seconds)')
ylabel('frequency (Hz)')
```