AMATH 482: Bops, Beats, and Banging: Classifying Music using SVD and LDA

Anh-Minh Nguyen

March 3, 2020

Abstract In this report, we will explore the classification of music using Linear Discriminant Analysis. The key will be identifying music and trying to guess the correct band or genre.

1 Introduction and Overview

Music is a collection of sounds that please our ears. It can paint emotions and speak a thousand words, literally and figuratively. Today, we are going to build upon a technique we used in our last reports, spectrograms and Singular Value Decomposition, and introduce Linear Discriminant Analysis to classify 5-second clips of music. We will have three different tests we will be running

- Test 1: We consider three different bands from three different genres and see if we can accurately identify a test set from the three chosen bands.
- Test 2: This is like Test 1, but we choose all three bands from the same genre. This might prove challenging for the test set this time.
- Test 3: We choose a variety of bands for each of three genres. In this case, we will see if we can identify the correct genre for our 5-second clips in the test set.

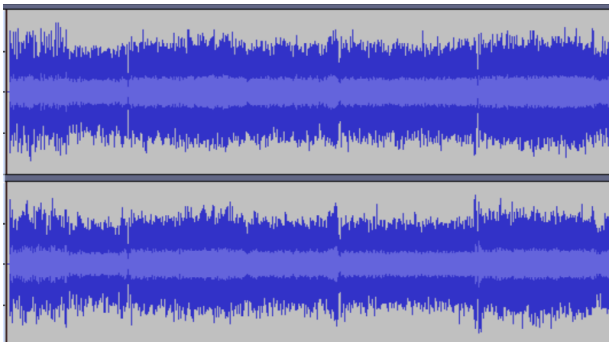

Figure 1: Sample Naruto Music

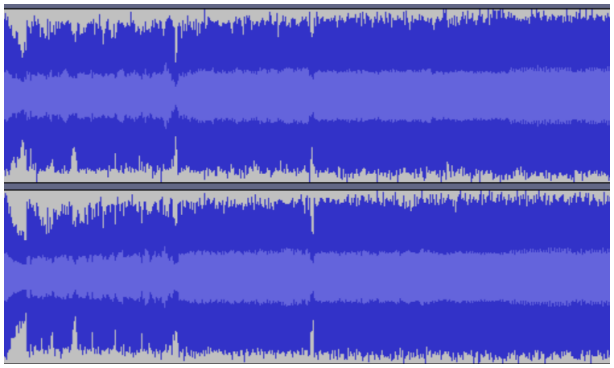

Figure 2: Sample Avicii Music

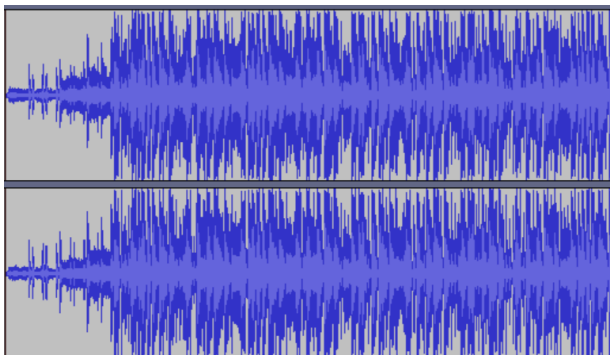

Figure 3: Sample Roddy Rich Music

For our Genres, I am going to use a collection of House songs, some of my favorite Utility Power Metal songs from the anime of manga I have read, and some 2018-2020 style hip hop music. Utility refers to fact that it's often used as background music.
For House, I have remixes of Avicii, The Chainsmokers, and Tiesto songs. The Utility Power Metal features Naruto, Dragonball Z: Budokai Tenkaichi 3, a video game actually, and One Punch Man. Finaly, we have songs from Lil Tjay, Polo G, and Roddy Rich.
So, let's turn it up and see how we are going to do this.

2 Theoretical Background

To begin, let's quickly refresh on a few topics. Spectrograms do have a visual component but they also, in MATLAB, perform short-time Fourier transform of an input signal. We will transform the signal, our 5-second clips of music, to identify key short-term, time-localized frequency content of each clip. We store each transformed signal in a matrix, A, making sure to note the position of the frequency to use the correct label.
We then use SVD, which can be used on our matrix A, to identify our principal projections. We then can, using a specified number of features, which help determine the amount of principal components we want. We then can project them onto our spectrogram matrix then separate the projections into our distinct groups.
The arguably most important step, or lesson, of all of this is LDA. The goal is to find an optimal projection onto a a new basis function that maximizes the distance between the inter-class groups while minimizing the intra-class groups.
We will be conducting a three-class LDA, so we consider the projection w,

$$w = \underset{w}{\operatorname{argmax}} \frac{w^T S_B w}{w^T S_W w} \tag{1}$$

where S_W is our within-class scatter matrix and S_B is our between-class scatter matrix. To reiterate, because this will be a three-class LDA, the equations are

$$S_B = \sum_{j=1}^{3} (\vec{\mu}_j - \vec{\mu})(\vec{\mu}_j - \vec{\mu})^T \tag{2}$$

$$S_W = \sum_{j=1}^{3} \sum_{\vec{x}} (\vec{x} - \vec{\mu}_j)(\vec{x} - \vec{\mu}_j)^T \tag{3}$$

They respectively measure the variances of differences within and outside the means of our data set. Using something called the generalized Rayleigh quotient:

$$S_B w = \lambda S_W w \tag{4}$$

We can then use eigenvalue decomposition to find the two eigenvectors with the largest absolute eigenvalues. We can then project the three level principal component projections onto our LDA eigenvectors. Unlike PCA, which maximizes the variance, LDA maximizes class-separation. Then, we will perform some KNN to to build a classifier model. It will label a certain point based on a k number of the nearest points labels.

3 Algorithm Implementation and Development

In the beginning, we start by applying the spectrogram to all our songs. I saved the songs in a particular order, so that we can use the index to keep track of each song. During this process, we grab twenty 5 second clips from each song. For every test, I used 15 songs, 3 songs for each required category. Afterwards, we can use *randperm* to generate a randomized sequence to help us separate the transformed clips into a training and test set, with a 50-50 split. Numerous rounds of randomization have given me some reason to split the data this way to minimize the model being over-fitted. Also, due to the limited number of songs I used for clips, my results are going to vary.

Next, we use *svd* to perform SVD on our training set, calculate the basis of principal projections and project on our training data. We do this by multiplying our matrix S with matrix V. Next, depending on the number of features we have chosen, we can form our scatter matrices for each of the categories.

Then we perform LDA, using for loops and the *eig*. First, we calculate the row means and grand mean then follow our formulas up above in the **Theoretical Background**. We do some fancy indexing to grab the two largest eigenvectors. The key is finding the index of the max eigenvalue, then we create a temporary matrix so we can remove the max value and then find the second max index.

After, we multiply them to our scatter matrices to find a threshold. From there, it's using the helpful commands, *fitcknn* and *predict* to build our model. I decided to use a k of 45. Again, after many iterations of testing, this mitigated the classification trap of classifying every clip in the test set as one of the three categories.

For our testing set, we do exactly the same thing, except we apply our model instead of building it. Now, let's how it went.

4 Computational Results

Let's begin at look at the percentage of results we got right for each Test. The table below

Table 1: Results Table: Best Numbers

	Training	Testing	Features
Test 1	.741-.972	.300-.800	35
Test 2	.831-.871	.300-.800	35-38
Test 3	.867-.988	.333-.647	38-42

As we can see, our results seem to indicate that we over-fitted our training data. However, I tested multiple training-test splits and **all** of them had approximately the same results. So, what I want to do is do look at my worst results.

(a) Test 1

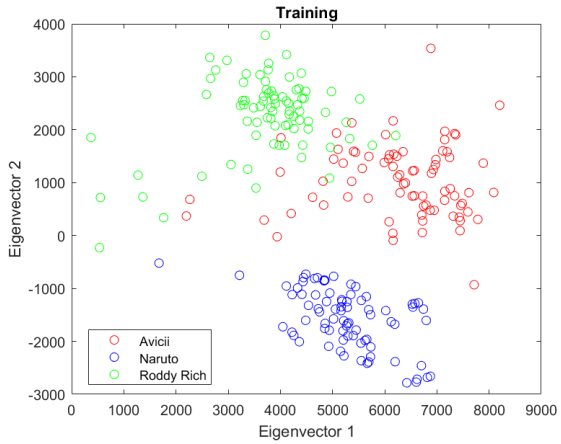

Figure 4: Train

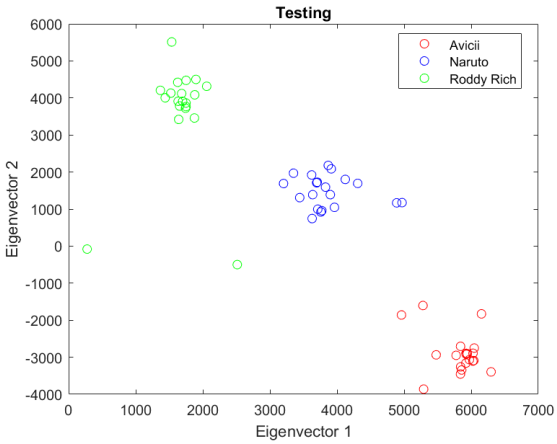

Figure 5: Test

Upon inspection, we see that our projections don't seem to match very well in the training and testing sets.

(b) Test 2

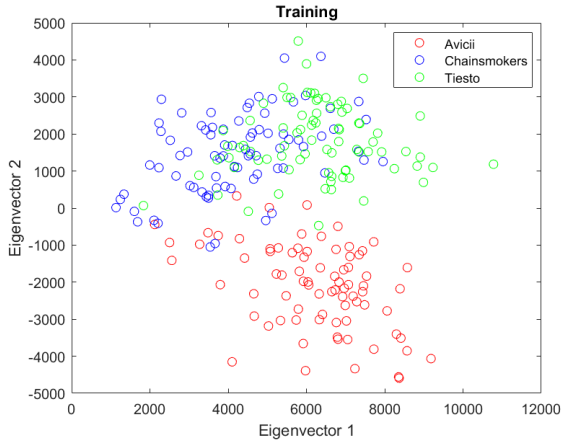

Figure 6: Train

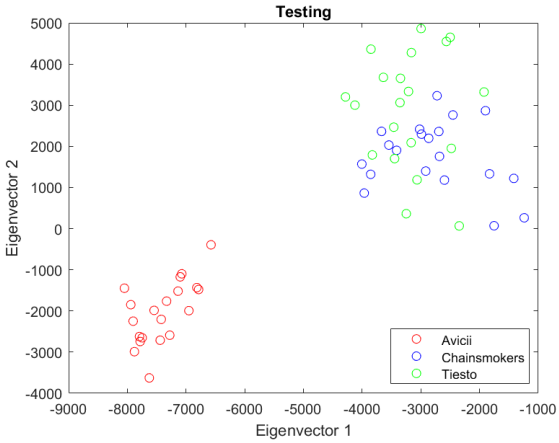

Figure 7: Test

In this case, the Testing set was off in it's own land based on the the maximum eigenvector, causing the model to incorrectly predict many of the clips.

(c) Test 3

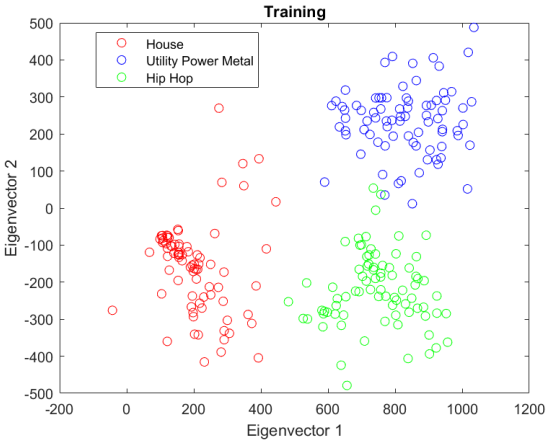

Figure 8: Train

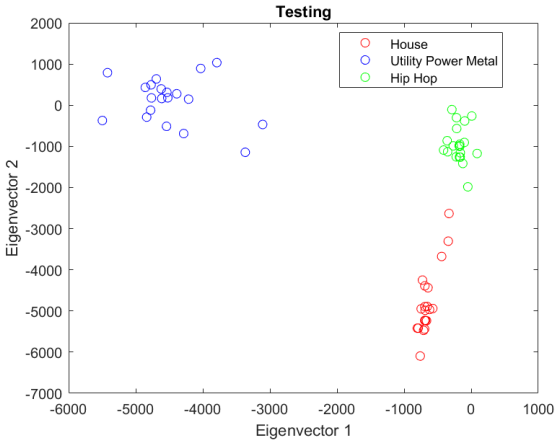

Figure 9: Test

Finally, like Test 2, Test 3's main eigenvector seems to be on another plot.
Overall, the computational results were too unpredictable, and the graphs are to help demonstrate issues I had.

5 Summary and Conclusions

Well, let's start with the elephant in the room: the predictability of the test results are awful. What the model is doing sometimes is labelling every new point as one of three classes. So, for every test, it will always get a third of the predictions right. However, that means it will always miss two thirds of the predictions as well. Sometimes, if the right mix of clips ended up in the training and test set, the algorithm would work great. A solution could have been to do k-fold cross validation and a more diverse pool of music clips.
We only had a reservoir of 5 songs per artist. So, the clips sometimes had different sections of the songs which could technically be from another genre outside of our chosen genres. For example, a soft RandB opening that leads into a harder hitting hip hop song. Furthermore, if we look at it closer, the three genres of music were not distinct enough possibly as well. But, that would mean it could possibly have a significantly higher percentage of correct predictions than 33%, which it did not have.
I eventually switched to a 50-50 split in hopes of increasing the randomness of the model. The training had no problem getting higher than a 70% accuracy rate, but the test accuracy was all over the place. Overall, I tried to build an algorithm that took in a random 50 clips every time, and occasionally I succeeded. But most of the time, I failed. One thing is that the second eigenvector seemed to be consistent in most cases.
We see how powerful LDA in maximizing the variance between means, but without a proper classification/thresholding method, we might not achieve the best results. I would like to say that I wasn't scared and chose three similar genres and artists within the genres.

6 Appendix

A MATLAB Functions

- *spectrogram*: Performs short-time, Fourier transform on an input signal.
- *eig*: Decomposes a matrix through eigenvalue decomposition.
- *svd*: Decomposes a matrix through singular value decomposition

- *fitcknn*: Fit a knn Model on a matrix of data
- *predict*: Requires a fitted model, then uses that model to predict new data points

B MATLAB Code

```
clear; close all; clc;

files = dir('music/Test 1/');
minsize = 193138; % Initial runs to find this number
full_data = zeros(300, 32769*8);
counter = 0;
split_data = randperm(100);
split_data2 = randperm(100);
split_data3 = randperm(100);
for file = files(3:end)'
    [y, Fs] = audioread(strcat('music/Test 1/', file.name));
    disp(file.name)
    y = mean(y, 2);
    for ii=1:20
        yy = y(Fs*(ii*5):Fs*(ii*5 + 5));
        full_data(counter*20 + ii,:) = reshape(abs(spectrogram(yy)), [1 32769*8])';
    end
    counter = counter + 1;
end
full_data = full_data';
X = full_data(:, [split_data (100 + split_data2) (200 + split_data3)]);
X2 = [X(:, 51:100) X(:, 151:200) X(:, 251:300)];
X = [X(:, 1:50) X(:, 101:150) X(:, 201:250)];
%% Test 1 Band Classification
% Naruto, Polo G, Avicii
% playblocking(audioplayer(yy,Fs))
an = 50; nn = 50; rn = 50;
[U,S,V] = svd(X,'econ');
%%
songs = S*V';
feature=30;
avicii = songs(1:feature,1:an);
naruto = songs(1:feature,an+1:an+nn);
roddyrich = songs(1:feature,an+nn+1:an+nn+rn);

mu = mean(songs(1:feature,:));
ma = mean(avicii,2);
mn = mean(naruto,2);
mr = mean(roddyrich,2);

%%
Sw=0;  % within class variances
for i=1:an
    Sw = Sw + (avicii(:,i)-ma)*(avicii(:,i)-ma)';
end
for i=1:nn
    Sw = Sw + (naruto(:,i)-mn)*(naruto(:,i)-mn)';
end
```

```
for i = 1:rn
    Sw = Sw + (roddyrich(:,i)-mr)*(roddyrich(:,i)-mr)';
end
Sb = (ma-mu)*(ma-mu)' + (mn-mu)*(mn-mu)' + (mr-mu)*(mr-mu)';
[V2,D] = eig(Sb,Sw);
[~,ind] = max(abs(diag(D)));
w=V2(:,ind); w=w/norm(w,2);

%%
D33 = D;
D33(:,ind) = [];
D33(ind, :) = [];
[~, indx] = max(abs(diag(D33)));
w1 = V2(:, indx + 1); w1 = w1/norm(w1,2);
%%
vavicii = w'*avicii; vnaruto = w'*naruto; vroddyrich = w'*roddyrich;
vavicii1 = w1'*avicii; vnaruto1 = w1'*naruto; vroddyrich1 = w1'*roddyrich;
w_basis = [w w1];
figure(1)
plot(vavicii, vavicii1, 'ro'); hold on;
plot(vnaruto, vnaruto1, 'bo');
plot(vroddyrich, vroddyrich1, 'go'); hold off;
title('Training')
legend("Avicii", 'Naruto', 'Roddy Rich','Location', 'best')
xlabel("Eigenvector 1")
ylabel("Eigenvector 2")
%%
results = [vavicii vnaruto vroddyrich];
res2 = [vavicii1 vnaruto1 vroddyrich1];
tabl = [results' res2'];
trueRes = [repelem("Avicii", 50) repelem("Naruto", 50) repelem("Roddy Rich", 50)]';
%%
Mdl = fitcknn(tabl,trueRes,'NumNeighbors',45,'Standardize',1);
label = predict(Mdl, tabl);
labelz = repelem("",150);
for hh = 1:150
   labelz(hh) = label{hh};
end
c = sum(trueRes==labelz')/150;

%% TEST SET
an2 = 50; nn2 = 50; rn2 = 50;
[U2,S2,V11] = svd(X2,'econ');
songs2 = S2*V11';
%%
feature=35;
avicii2 = songs2(1:feature,1:an2);
naruto2 = songs2(1:feature,an2+1:an2+nn2);
roddyrich2 = songs2(1:feature,an2+nn2+1:an2+nn2+rn2);

mu2 = mean(songs2(1:feature,:));
ma2 = mean(avicii2,2);
mn2 = mean(naruto2,2);
mr2 = mean(roddyrich2,2);
```

```
%%
Sw2=0;  % within class variances
for i=1:an2
    Sw2 = Sw2 + (avicii2(:,i)-ma2)*(avicii2(:,i)-ma2)';
end
for i=1:nn2
    Sw2 = Sw2 + (naruto2(:,i)-mn2)*(naruto2(:,i)-mn2)';
end
for i = 1:rn2
    Sw2 = Sw2 + (roddyrich2(:,i)-mr2)*(roddyrich2(:,i)-mr2)';
end
Sb2 = (ma2-mu2)*(ma2-mu2)' + (mn2-mu2)*(mn2-mu2)' + (mr2-mu2)*(mr2-mu2)';
[V22,D2] = eig(Sb2,Sw2);
[lambda,ind2] = max(abs(diag(D2)));
w2=V22(:,ind2); w2=w2/norm(w2,2);
%%
D332 = D2;
D332(:,ind2) = [];
D332(ind2, :) = [];
[~, indx2] = max(abs(diag(D332)));
w12 = V22(:, indx2 + 1); w12 = w12/norm(w12,2);
%%
vavicii2 = w2'*avicii2; vnaruto2 = w2'*naruto2; vroddyrich2 = w2'*roddyrich2;
vavicii12 = w12'*avicii2; vnaruto12 = w12'*naruto2; vroddyrich12 = w12'*roddyrich2;
results2 = [vavicii2 vnaruto2 vroddyrich2];
res22 = [vavicii12 vnaruto12 vroddyrich12];
tabl2 = [results2' res22'];
trueRes2 = [repelem("Avicii", 50) repelem("Naruto", 50) repelem("Roddy Rich", 50)]';
%%
label2 = predict(Mdl, tabl2);
labelz2 = repelem("",150);
for hh = 1:150
   labelz2(hh) = label2{hh};
end
c2 = sum(trueRes2==labelz2')/150;
%%
figure(2)
plot(vavicii2, vavicii12, 'ro'); hold on;
plot(vnaruto2, vnaruto12, 'bo');
plot(vroddyrich2, vroddyrich12, 'go'); hold off;
title('Testing')
legend("Avicii", 'Naruto', 'Roddy Rich','Location', 'best')
xlabel("Eigenvector 1")
ylabel("Eigenvector 2")
```

The other parts are exactly the same, except for a few name changes.

C References

MATH 482 Course Notes