Homework 4: Music Classification

Maximilian Walter
AMATH 482
University of Washington
3/6/2020

The goal of this assignment is to build a classifier that can determine different types of music based on genre and artist using short audio clips. To build this model I used linear discriminant analysis (LDA) and primary component analysis (PCA) to analyze and create thresholds between different clips of music. Overall, the model performed poorly, especially when attempting to distinguish between genres.

Introduction and Overview

The first piece of this assignment was to build a model that can classify three different artists. For this, I used 5-second clips from ten different songs each by Five Finger Death Punch, Odesza, and Ludwig van Beethoven. Each clip was recorded from Spotify for the use in this assignment only and the clips will not be distributed. Each clip was recorded at 44.1kHz and then turned into a spectrogram for better analysis. Each clip was then decomposed using PCA and a threshold between each category was determined using LDA. For each artist 40 clips were used as training sets and 10 clips for testing, these were chosen at random from the beginning.

For the second piece of the assignment, the same model was used to try and classify three artists within the same genre. For this, I picked Metal as my genre. I used songs from Five Finger Death Punch, Metallica, and Slipknot for this experiment. The same process of converting to a spectrogram was done to each clip and then run through PCA and LDA to create the model.

The final model was the more general case in which the model was used to classify genres from a given song clip. The model was trained to distinguish between Pop, Classical Music, and Metal. For this run, I used a training set of 120 clips for each genre and a test set of 30 clips per genre.

Theoretical Background

SVD and PCA

Singular Value Decomposition is a process of manipulating a matrix, A, into three components [U, S, V]. Each component contains information about the original matrix. V contains the shape of our data, S then stretches or shrinks the data, and U rotates it to a new axis. The idea behind this is to collapse the data's dimensions and shows the variance within the dataset. After completing the SVD of our data set, we can then project this data onto a new basis using Primary Component Analysis (PCA). This reduces the number of dimensions needed to explain the data and allows us to group the data within this new basis. Using LDA, we are then able to optimize which basis best separates the data for better classification.

LDA

Linear Discriminant Analysis is used to find which projected basis is best suited to separate the data by maximizing the distance between inter-class data and minimizing the intra-class data. This is done by creating a projection w, the uses the scatter matrices between classes, S_B, and within-class, S_W. The equation for this projection is given in eq1.

$$w = max\frac{w^T S_B w}{w^T S_W w} \tag{1.1}$$

The equations for S_B and S_W are as follows

$$S_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T \tag{1.2}$$
$$S_W = \sum_{j=1}^{2}\sum(x - \mu_j)(x - \mu_j)^T \tag{1.3}$$

The final step to finding the projection basis of interest is to solve the generalized eigenvectors for equation 2.

$$S_B w = \lambda S_W w \tag{2}$$

Solving these equations allows us to create a basis that maximizes the separation of data and thus allows us to create thresholds between clusters of data for later classification.

Algorithm Development

Audio Clips and Spectrograms

The first piece of this assignment was to get the audio clips to train and test the model. This was done using QuickTime Player and Spotify to record each song by routing the audio directly into the recording software. After each clip was obtained, I created a MATLAB function that would then load and convert these files into a spectrogram. This was done using the built-in MATLAB command, *spectrogram.* The function would loop through each recording, converting 5 seconds of the .m4a file to a spectrogram, and storing the results as a single column vector. This was then concatenated into one large matrix and exported as a .csv file to the folder. The function can be found in Appendix B.

SVD and PCA

After each clip was turned into a spectrogram, I created a second function which would serve as the trainer. The function takes in an input of the training data from each artist or genre, and the number of features, or primary components, the user would like to keep. The data from each artist would then be turned into a single matrix and then would undergo SVD. For improved efficiency, the "economic" version of the SVD command was used. From here, each artist's data would then be recreated using PCA to project the data onto a new vector.

LDA

After the data was recreated using PCA, LDA was used to find the threshold between the different sets of data. The function loops through each artist and concatenated the data into the S_W and S_B vectors. The eigenvalues of these vectors are then found to create the W vector, which the 2-norm. This allows us to find the variation between each artist. A while loop is then used to comb through the data until the minimum variation is found which determines the threshold between the artists. These thresholds are then exported out of the function along with the U, S, V components.

Running the Model

After creating the proper model, the final script pulls all these functions together. Initially, I used the first 40 clips to train the data and the final 10 clips to test the data, but since the data was taking sequentially from songs, the final 10 test cases would all be from the same song. Therefore, I small piece of code that randomly assigns 10 clips to be test data and the other 40 clips to be training data. This allowed for variation and better testing of the model. This was then fed into the training model and the output is the U, S, V matrices, the variance matrix, and the thresholds found. Using the test data, the variance matrix was used to create a set of values that can be compared to the thresholds. These were labeled as such. These results were then compared to a vector of the true labels to determine if the model correctly classified each clip of music. This was done using logic gates and the answers were then summed, which would output how many times the model was correct. This value, divided by the total number of clips tested, gave the accuracy of each run.

Computational Results

The first experiment comparing different artists resulted and an accuracy ranging between 40 and 63%. This is due to the randomization of both test and training data. Below is an image showing the different projections of the artists as well as the thresholds used to classify them. The data has been staggered for better visualization.

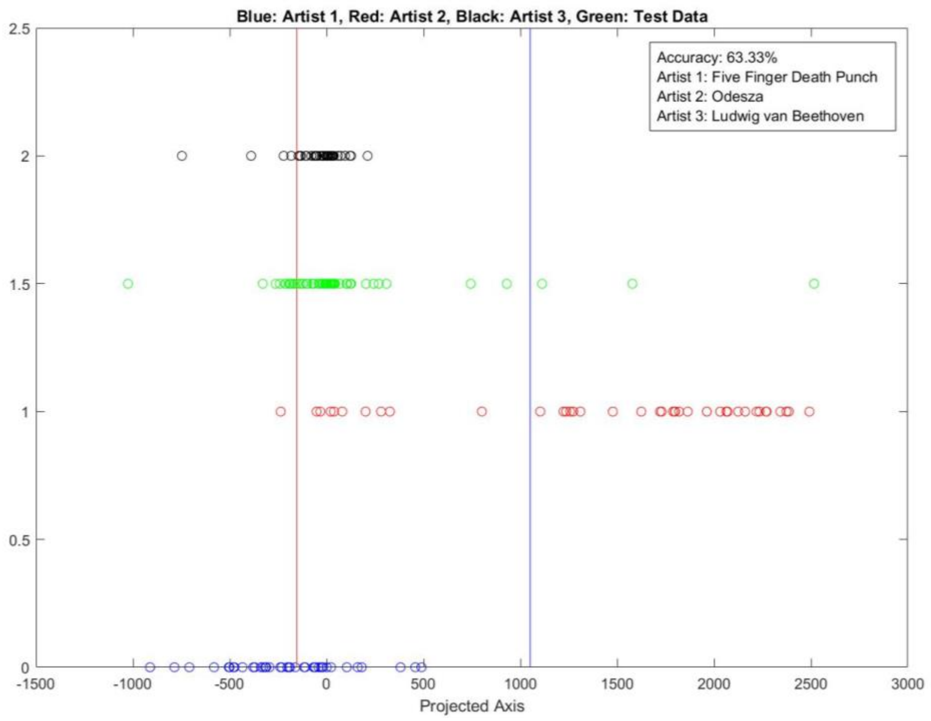

Figure 1. Projected training data are shown in blue, red, and black. Test data are shown in green. Thresholds are shown as vertical lines that were used to classify each category. The accuracy of this run was 63.33%.

For the second part of the assignment, classifying different artists from within the same genre, the model did worse than with three distinct artists. Using Metal as the genre for this run, I used songs from Slipknot, Five Finger Death Punch, and Metallica. the best result is shown below with an accuracy of 66.67%.

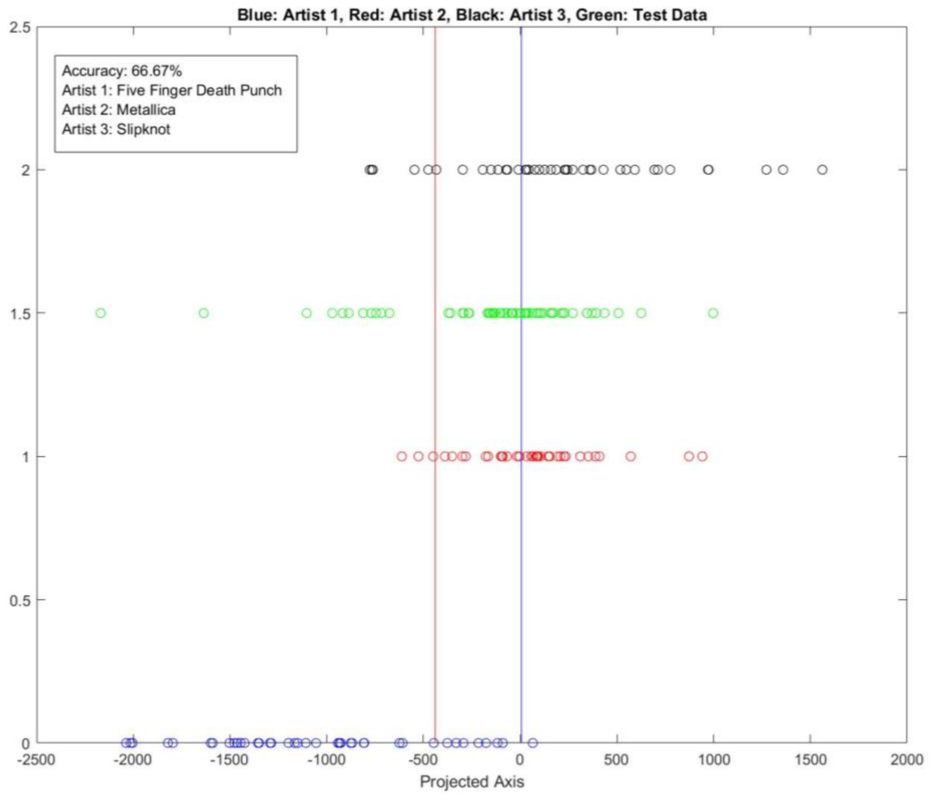

Figure 2. The training data is shown in blue, red, and black. Testing data shown in green. Thresholds used to classify data shown on the graph, this resulted in an accuracy of 66.67%.

The final assignment, classifying genre resulted in the graph below. The best run resulted in an accuracy of 53.33%.

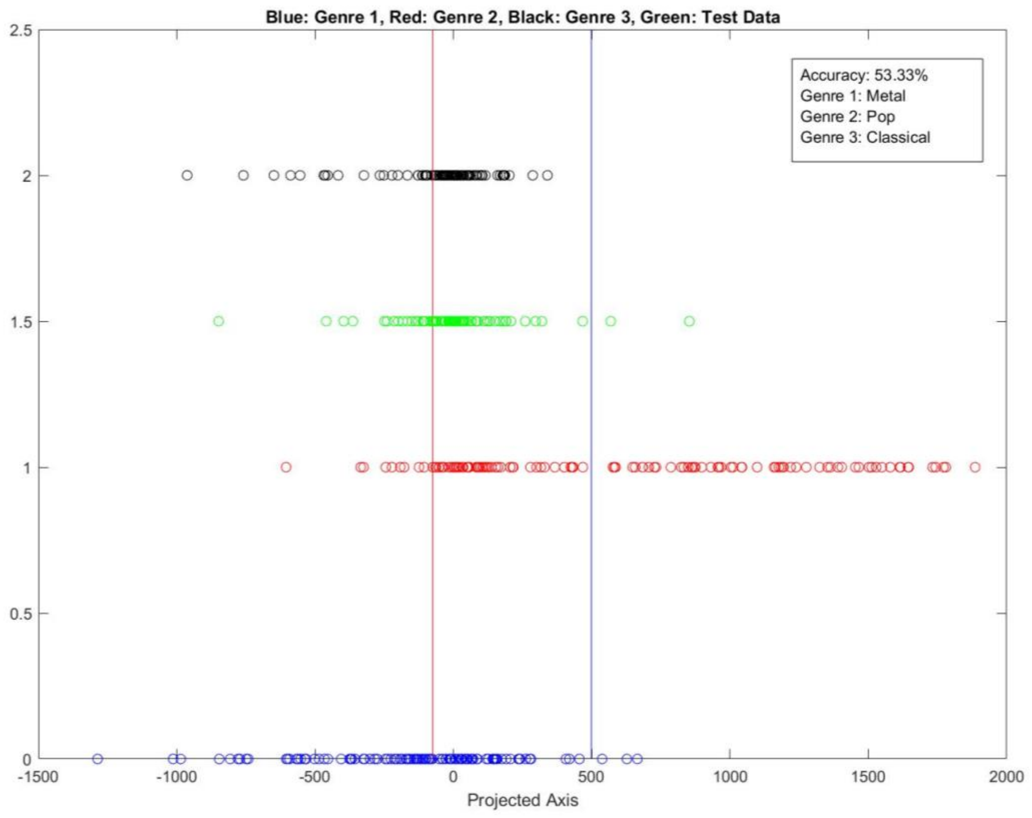

Figure 3. Classification of genres. Metal, Pop, and Classical music genres were used to train this model. Thresholds are shown on the graph with a resulting accuracy of 53.33%.

Summary and Conclusion

Overall, this model was not extremely effective. I was able to build a classifier that can take input spectrograms, reduce the data into primary components, and establish thresholds using LDA. The overall model resulted in scores better than guess, 33.3%, but still fell short of 2/3 or higher chance of choosing the correct genre or artists. The model also required using almost all of the features to create a respectable amount of data, which would not scale well to large data sets, or a model with more than three variables. I believe filtering in the audio files could improve the model. Creating filters that only evaluate the artist's vocals or specific instruments in each song would help create specific “edges” for the model to learn from. Currently, as seen in the spectrograms, there are no distinct pieces that the model can use to say this is X artists or genre.

Appendix A

SVD(x,'econ'): This function takes the SVD of the input matrix A. For computational speed, the economic version was taken which removes zeros from the matrix.

[V, D] = Eig(X, Y): takes the generalized eigenvalue of the vectors X and Y and returns the diagonal and the right eigenvectors.
Norm(x,2): returns the 2-norm of x.

Z = Z(Z~=A): This removes any occurrence of A within the vector or matrix Z. This is used to remove the random variables chosen to be testing data from the training set.

Appendix B

Function 1: Audio to Spectrogram

```
%AMATH 482-HW 4
%Audio to Spec File 2

data_path = 'C:\Users\maxim\MATLAB_DRIVE\Class\AMATH\AMATH_482\HW_4';
audio_file = '\Audio_Files2';
artist = '\JSB';
prefix = '\JSB_';
ext = '.m4a';
FS = 44100;
spec_out = [];
step = 0;
run = 0;

for j = 1:10
    clear data
    dir = [data_path audio_file artist prefix num2str(j) ext];
    data = audioread(dir);
    %data = downsample(data,10);
    data = data';
    leng = FS * 5;
    step = 0;
    for i = 1:5
        file = data(1,(step + 1):(i*leng));
        spec = spectrogram(file);
        spec_out(:,i + run) = spec(:);
        step = step + leng;
    end
    run = run + 5;
end

writematrix(spec_out,'JSB.csv')
```

Function 2: Training the model using SVD, PCA, and LDA

```
function [result,w,U,S,V,threshold1,threshold2] =
artist_trianer3(t_art1,t_art2,t_art3,feature)

nart1 = length(t_art1(1,:));
```

```
nart2 = length(t_art2(1,:));
nart3 = length(t_art3(1,:));


[U,S,V] = svd([t_art1 t_art2 t_art3],'econ');

artists = S*V';
U = U(:, 5:feature);

t_art1 = artists(1:feature,1:nart1);
t_art2 = artists(1:feature,nart1+1:nart1+nart2);
t_art3 = artists(1:feature,(nart1+nart2)+1:nart1+nart2+nart3);
art_all = [t_art1 t_art2 t_art3];

mart1 = mean(t_art1,2);
mart2 = mean(t_art2,2);
mart3 = mean(t_art3,2);
mean_all = mean(art_all,2);

Sw = 0;

for i = 1:nart1
    Sw = Sw + (t_art1(:,i)-mart1)*(t_art1(:,i)-mart1)';
end

for i = 1:nart2
    Sw = Sw + (t_art2(:,i)-mart2)*(t_art2(:,i)-mart2)';
end

for i = 1:nart3
    Sw = Sw + (t_art3(:,i)-mart3)*(t_art3(:,i)-mart3)';
end
Sb = (mart1 - mean_all)*(mart1 - mean_all)' + (mart2 - mean_all)*(mart2 -
mean_all)' + (mart3 - mean_all)*(mart3 - mean_all)';
Sb = Sb/3;

[V2,D] = eig(Sb,Sw);
[lambda,ind] = max(abs(diag(D)));
w = V2(:,ind);
w = w/norm(w,2);

vart1 = w'*t_art1;
vart2 = w'*t_art2;
vart3 = w'*t_art3;
result = [vart1,vart2,vart3];

sortart1 = sort(vart1);
sortart2 = sort(vart2);
sortart3 = sort(vart3);

t1 = length(sortart1);
t2 = 1;

while sortart1(t1)>sortart3(t2)
```

```
    t1 = t1 - 1;
    t2 = t2+1;

end
threshold1 = (sortart3(t1)+sortart1(t2))/2;

t2 = length(sortart1);
t3 = 1;

while sortart1(t2)>sortart2(t3)
    t2 = t2 - 1;
    t3 = t3+1;
end

threshold2 = (sortart2(t2)+sortart3(t3))/2;

plot(sortart1,0,'bo')
hold on
plot(sortart2,1,'ro')
plot(sortart3,2,'ko')
title('Blue: Artist 1, Red: Artist 2, Black: Artist 3, Green: Test Data')
xlabel('Projected Axis');
plot([threshold1 threshold1],[0 2.5],'r')
plot([threshold2 threshold2],[0 2.5],'b')
```

Function 3: Quantifying accuracy using test data.

```
clear all; close all; clc
%%
art1 = load('FFDP.csv');
%%
art2 = load('MA.csv');
%%
art3 = load('SK.csv');
%%
close all;
tot = 1:50;
P = randperm(50,20);
P = sort(P);
% P = 1:5:50;
for i = 1:10
    tot = tot(tot~=P(i));
end
% tot = [1:40];
% P = 41:50;

train_art1 = art1(:,tot);
test_art1 = art1(:,P);

train_art2 = art2(:,tot);
test_art2 = art2(:,P);

train_art3 = art3(:,tot);
test_art3 = art3(:,P);
```

```
close all;
feature = 40;
[result,w,U,S,V,threshold1,threshold2] =
artist_trianer3(train_art1,train_art2,train_art3,feature);
%[result,w,U,S,V,threshold] = artist_trianer2(train_art1,train_art3,feature);
test_set = [test_art1 test_art2 test_art3];

TestMat = U'*test_set;
pval = w(5:feature)'*TestMat;
pval = sort(pval);

label = zeros(30,1);
for i = 1:30
    if pval(1,i) < threshold1
        label(i) = 1;
    elseif pval(1,i) > threshold1 && pval(1,i) < threshold2
        label(i) = 3;
    else
        label(i) = 2;
    end
end

hidden = zeros(30,1);
hidden(1:10) = 1;
hidden(11:20) = 2;
hidden(21:30) = 3;
total = 30;

correct = zeros(30,1);
for i = 1:30
    if label(i) == hidden(i)
        correct(i) = 1;
    else
        correct(i) = 0;
    end
end

accuracy = sum(correct)/total;
error = 1 - accuracy;
hold on
plot(pval,1.5,'go')
```